Cheats & mistakes to read and create SLAs

Vasily Pantyukhin

High Load ++ Armenia

Co-organizer

Yandex

VeUP The Platform for ISV Growth

We make you more attractive to investors

a new kind of consultancy

- The only AWS VAR exclusively working with ISVs
- Reduce AWS billing challenges
- Technical & GTM consultative services at a reduced or zero cost
- Get investment from AWS & external investors

Promise

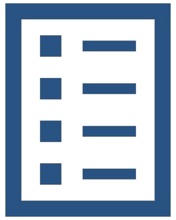
&

Trust

SLI

Service-Level Indicator - a quantitative metric used to measure the compliance with an SLO

- *Availability, Latency, Performance, Error rates, Utilization, etc*



SLO

Service Level Objective – a target SLI value or range of values within the SLA

- *SLI ≤ target*
- *lower bound ≤ SLI ≤ upper bound*

SLA

Service Level Agreement – an agreement between the service provider and the client

- *level of the services to be delivered*
- *service measurement metrics*
- *penalties if the expected level of services is not met*

Catch me if you can

tunefind.com/movie/catch-me-if-you-can-2002

- 😈 Cheats
- ☹ Mistakes
- 😀 Tips

Read from the end

Here and further, “Catch me if you can” movie screenshots are used
as Fair Use for non-profit educational purposes (Copyright Act of 1976)

Be precise with ALL definitions

mazon SQS and SNS example

SLI metric

Definitions

- "Availability" is calculated for each 5-minute interval as the percentage of Requests processed by the applicable Included Service that do not fail with Errors and relate solely to the provisioned Included Service. If you did not make any Requests in a given 5-minute interval, that interval is assumed to be 100% available.
- An "Error" is any Request that returns a 500 or 503 error code.
- "Monthly Uptime Percentage" for a given AWS region is calculated as the average of the Availability for all 5-minute intervals in a monthly billing cycle. Monthly Uptime Percentage measurements exclude downtime resulting directly or indirectly from any Messaging SLA Exclusion (defined above).
- A "Request" is, with respect to:
  - SNS: an API request to SNS by directly calling the Publish API or triggered by a supported event source; and
  - SQS: invocation of a SQS Send, Receive, or Delete API.
- A "Service Credit" is a dollar credit, calculated as set forth above, that we may credit back to an eligible account.

How SLI is measured

Use “intuition”

Always define common terms like “request”, “failure”, etc

alOcean Kubernetes example

Service Commitment

DigitalOcean Kubernetes (DOKS) provides 99.95% uptime SLA per month for the control plane when high availability (HA) is enabled for such clusters. The SLA is effective on the billing period starting July 1, 2022.

Definitions

The terms used in this SLA document are defined as follows:

- **Monthly Uptime**: For a given DOKS HA Cluster, monthly uptime is calculated by subtracting from 100% the percentage of 5-minute intervals during the monthly billing cycle in which the DOKS Cluster control plane was Unavailable. If the DOKS cluster exists for only part of the month, availability is calculated over the portion of the month during which it existed. Monthly Uptime measurements exclude Unavailability resulting directly or indirectly from any SLA exclusion.
- **Service Credit**: Credit in terms of $USD issued to the associated DigitalOcean account.
- **Unavailability**: All the requests to the DOKS HA control plane of a cluster fail for more than 5 minutes.

?

Always define common terms like “request,” “failure,” etc

alOcean Kubernetes example

Service Commitment

DigitalOcean Kubernetes (DOKS) provides 99.95% uptime SLA per month for the control plane when high availability (HA) is enabled for such clusters. The SLA is effective on the billing period starting July 1, 2022.

Definitions

The terms used in this SLA document are defined as follows:

- **Monthly Uptime**: For a given DOKS HA Cluster, monthly uptime is calculated by subtracting from 100% the percentage of 5-minute intervals during the monthly billing cycle in which the DOKS Cluster control plane was Unavailable. If the DOKS cluster exists for only part of the month, availability is calculated over the portion of the month during which it existed. Monthly Uptime measurements exclude Unavailability resulting directly or indirectly from any SLA exclusion.
- **Service Credit**: Credit in terms of $USD issued to the associated DigitalOcean account.
- **Unavailability**: All the requests to the DOKS HA control plane of a cluster fail for more than 5 minutes.

What if “all requests” fail with 4*xx* error code?


Share

Share "designed for" SLO

Amazon S3 example

	S3 Standard	S3 Intelligent-Tiering*	S3 Standard-IA	S3 One Zone-IA†	S3 Glacier Instant Retrieval	S3 Glacier Flexible Retrieval
Designed for durability	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)
Designed for availability	99.99%	99.9%	99.9%	99.5%	99.9%	99.99%
Availability SLA	99.9%	99%	99%	99%	99%	99.%
Availability Zones	≥3	≥3	≥3	1	≥3	≥3

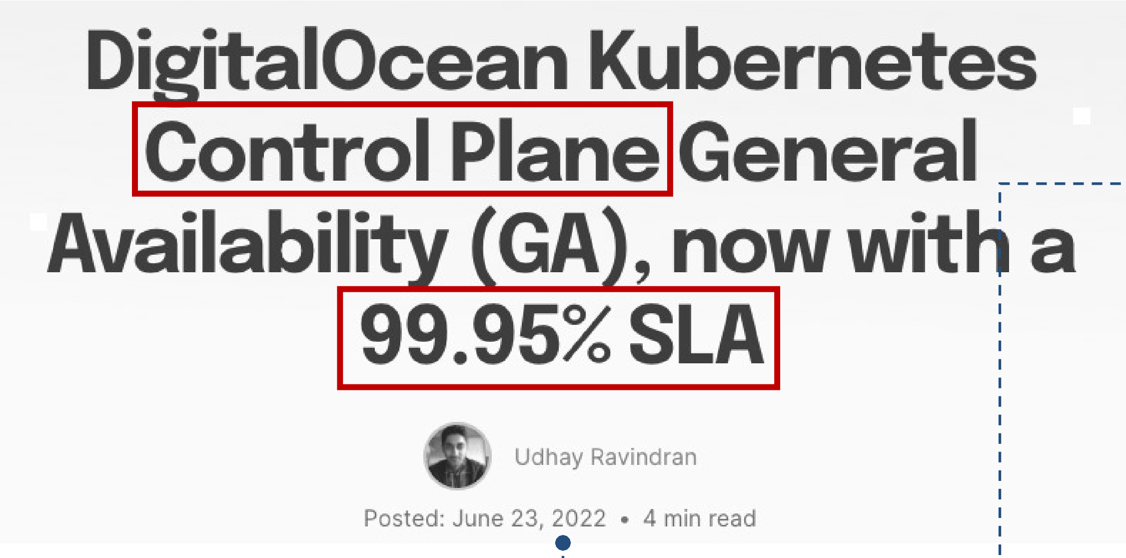
SLO

SLA

Formal SLA is stricter than "designed for" SLO

Share Control Plane SLA/SLO

gitalOcean Kubernetes and Amazon SQS/SNS examples



DigitalOcean Kubernetes Control Plane General Availability (GA), now with a 99.95% SLA

Udhay Ravindran

Posted: June 23, 2022 • 4 min read

Control Plane SLA

Data Plane SLO is stricter

Service	Component	Availability Design Goal
Amazon Simple Notification Service (Amazon SNS)	Data Plane	99.990%
	Control Plane	99.900%
Amazon Simple Queue Service (Amazon SQS)	Data Plane	99.980%
	Control Plane	99.900%

Separate single instance and Multi-zonal SLAs

gle Compute Engine example

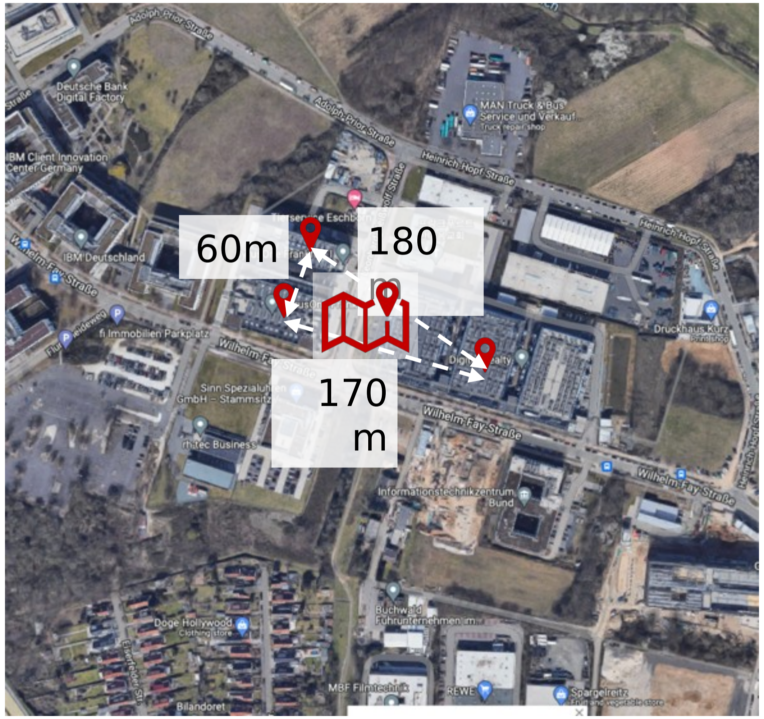
Covered Service	Monthly Uptime Percentage
Instances in Multiple Zones	>= 99.99%
A Single Instance	>= 99.5%

Multi-zonal vs. Instance level SLAs

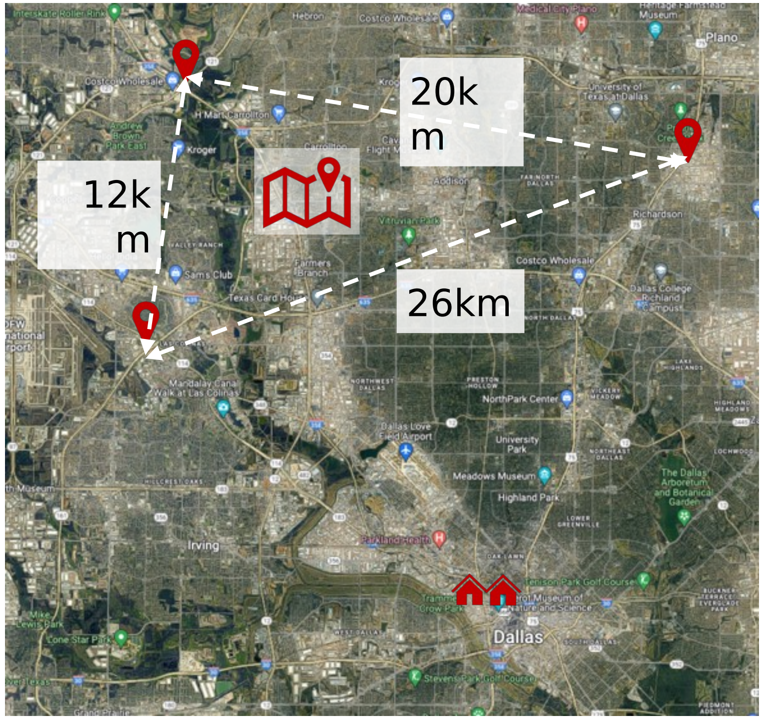

Overstate

C = Zone, Town = Region

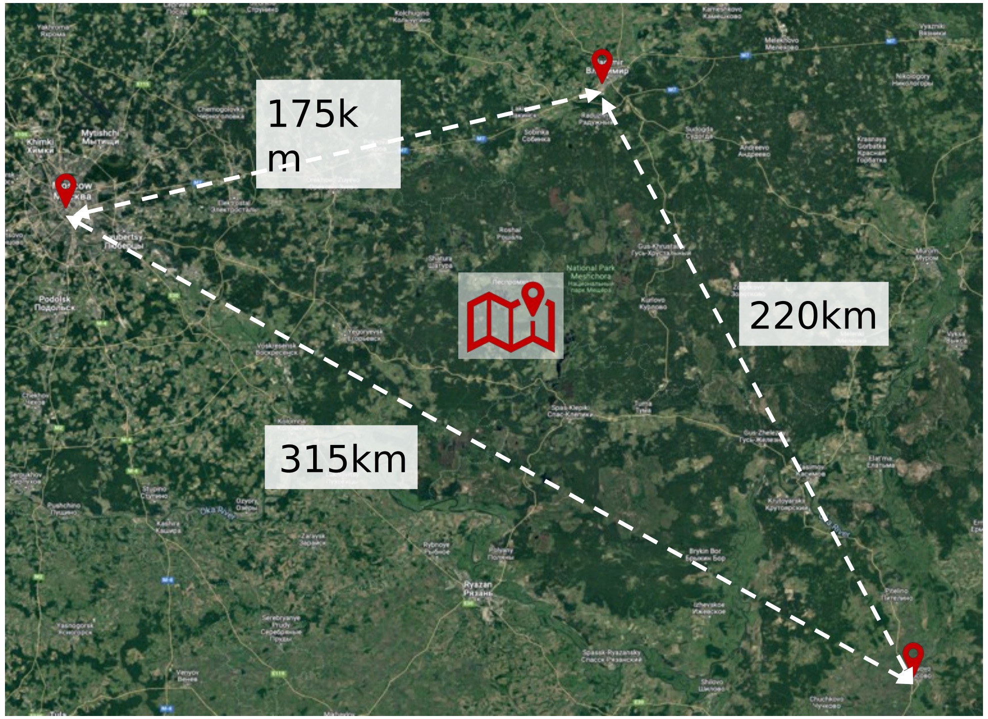
Vendor 1



Vendor 2



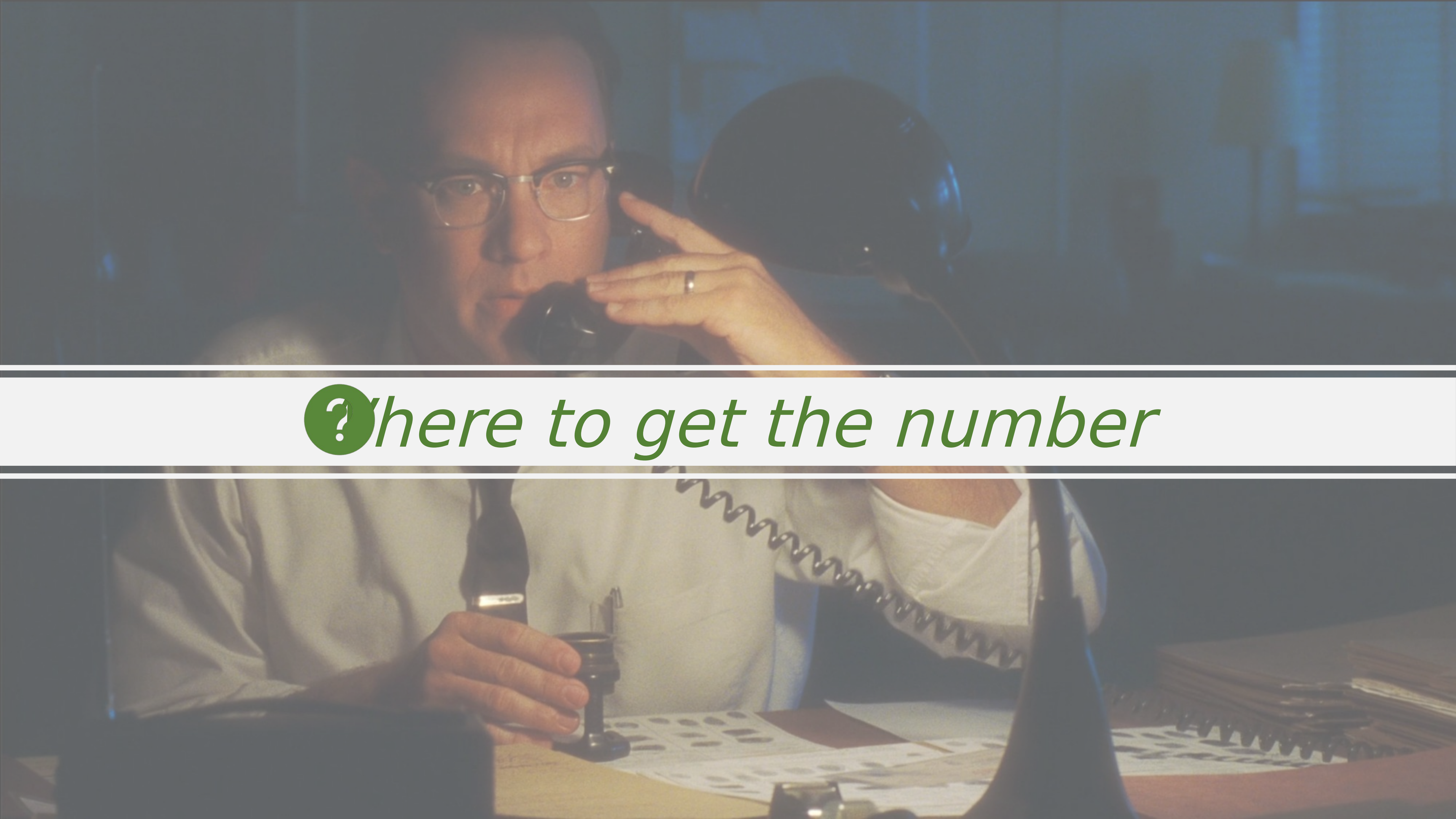
Vendor 3



Standalone DC

Availability Zone

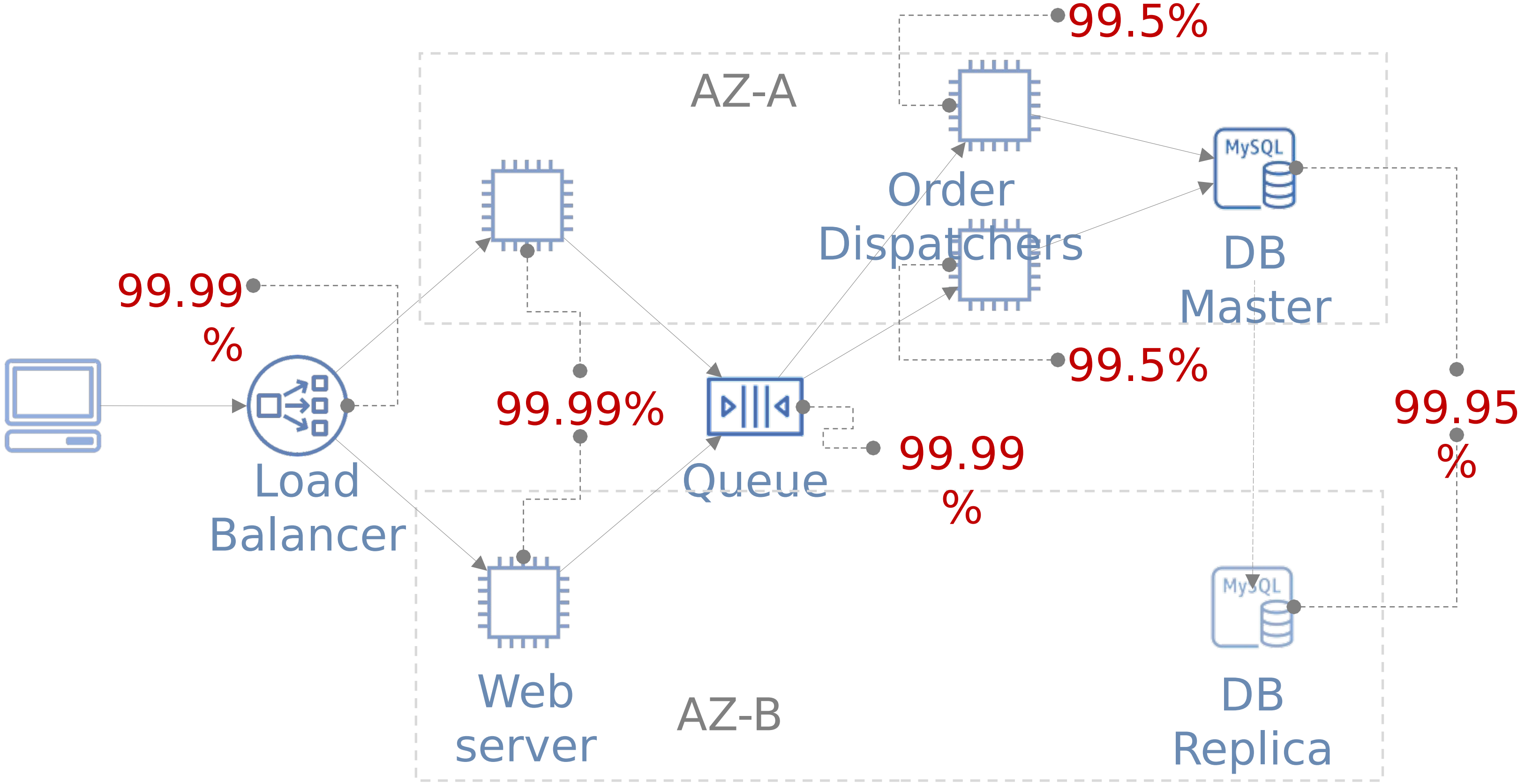
Multi-Zonal Region

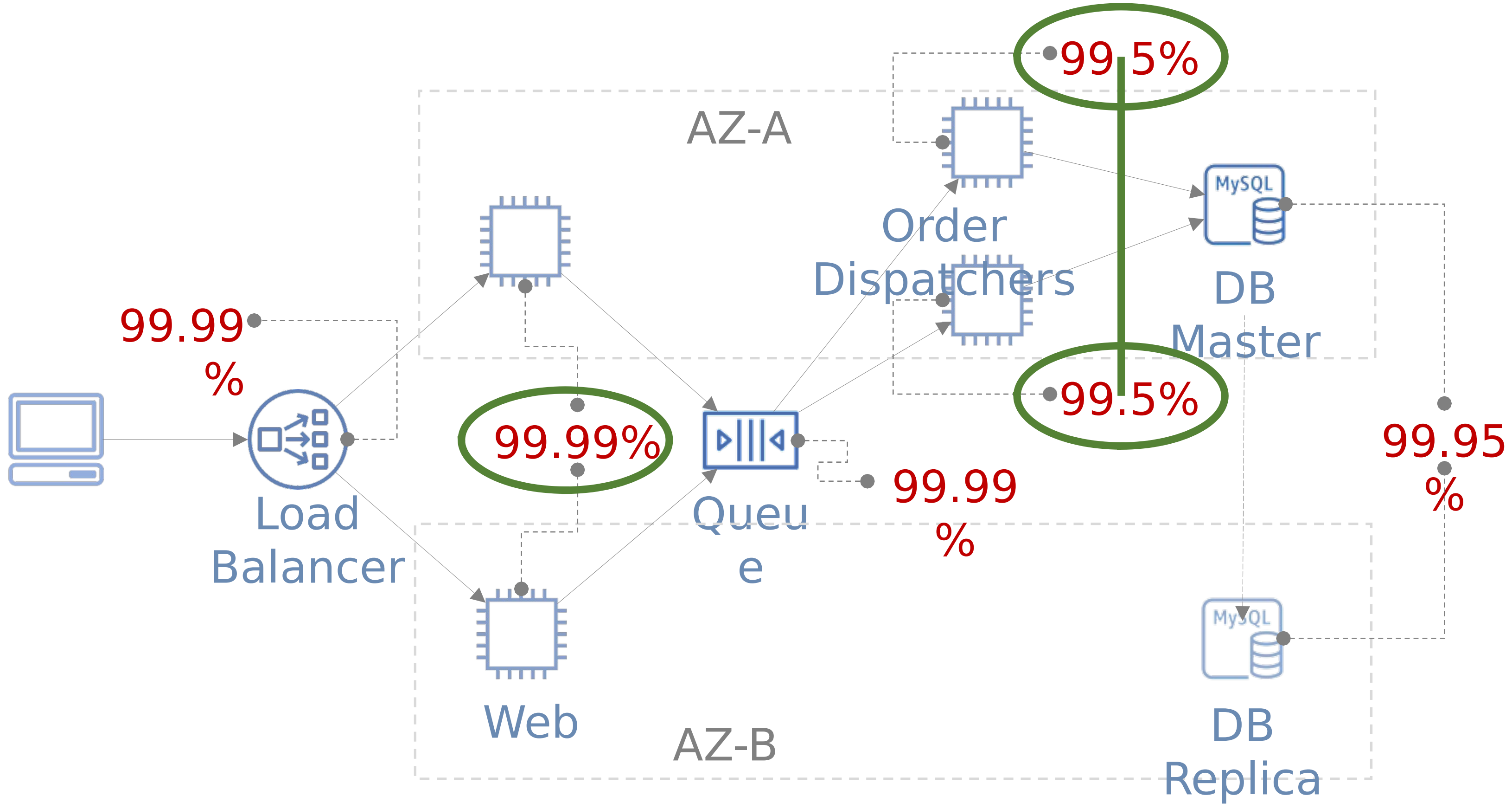

Where to get the number

Reliability

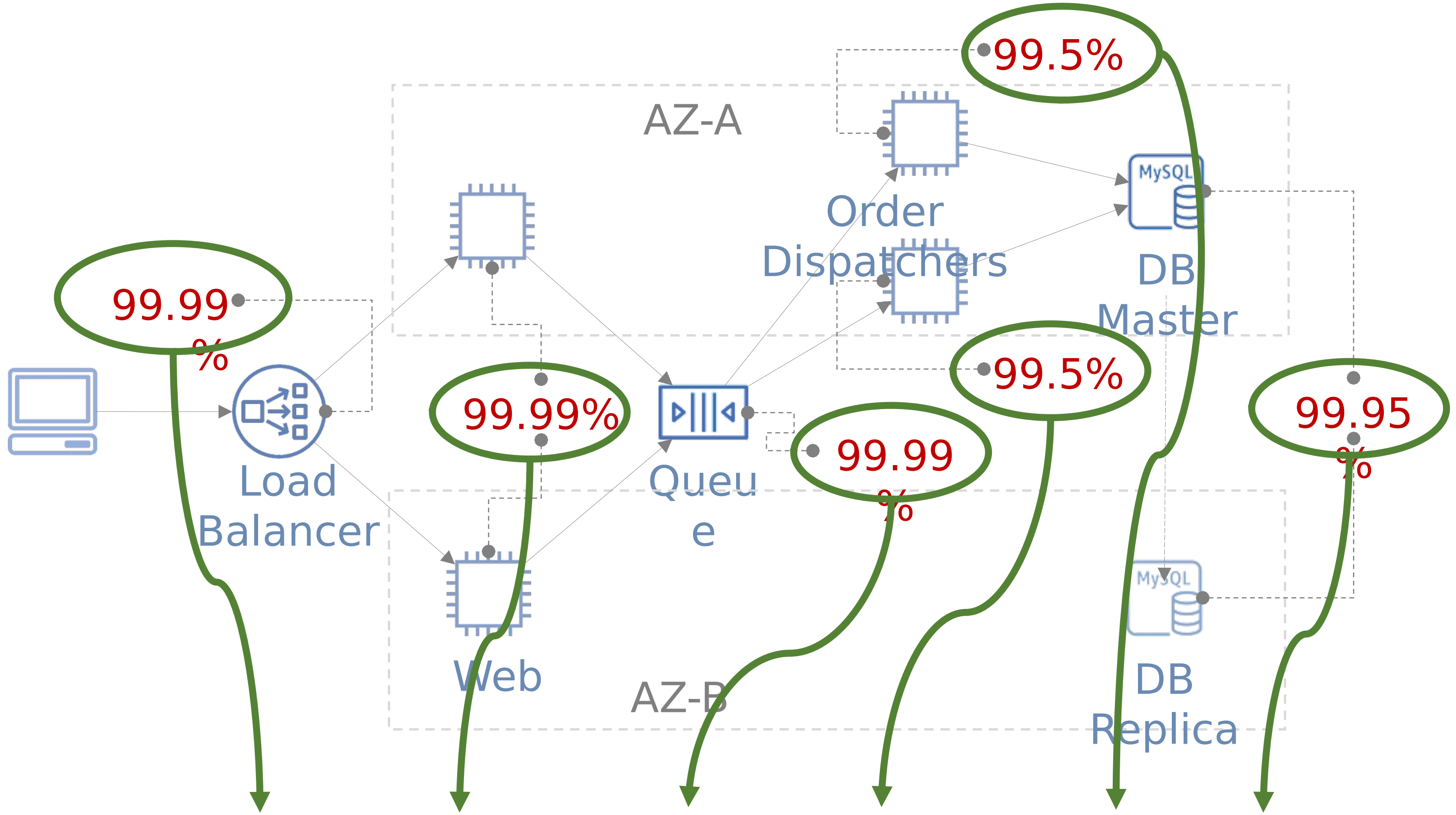
	Availability	**Durability**
Objectives	Maximizing uptime	Minimizing suffer from data loss and corruption
Focus on	Components of the system	Data the system works with
Time	Mean values over a mid-term period	Long-term likelihood of data loss or corruption
Scale	Small-scale common disruptions	Small-scale corruption and large-scale loss problems

Example architecture





99.5%
AZ-A
Order
Dispatchers
DB
Master
99.99
%
99.99%
99.5%
99.95
%
99.99
%
Load
Balancer
Queu
e
Web
AZ-B
DB
Replica

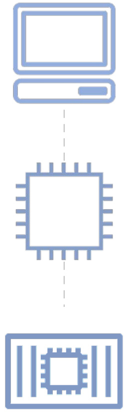


bility = min(0.9999, 0.9999, 0.9999, 0.995, 0.995, 0.9995) = 99.5%


☹Take min

Availability with Dependencies

$$\boxed{Availability_{system} = A_1 * A_2}$$



A_1=99%

A_2=99.5%

$$Availability_{system} = 0.99 * 0.995 = 98.5\%$$

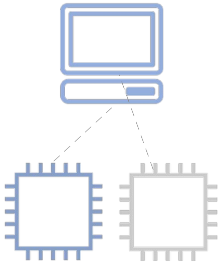
↓Hard dependencies = ↑Availability

Hard - workload cannot function without it
Soft - unavailability can go unnoticed or tolerated for some period of time

$A_{1,2}$ – Availability of components

Availability with full Redundancy

$$\boxed{Availability_{system} = 1 - (1 - A_1) * (1 - A_2)}$$



A_1=99 % A_2=99 %

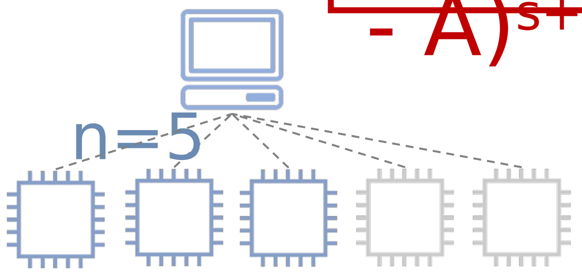
$Availability_{system} = 1-(1-0.99) * (1-0.99) = 99.99\%$

↑Redundancy = ↑Availability

A1,2 – Availability of components

Availability with partial Redundancy

$$\boxed{Availability_{system} \approx 1 - f * (1 - A)^{s+1} \qquad f = \frac{n!}{(s+1)! * (n-s-1)!}}$$



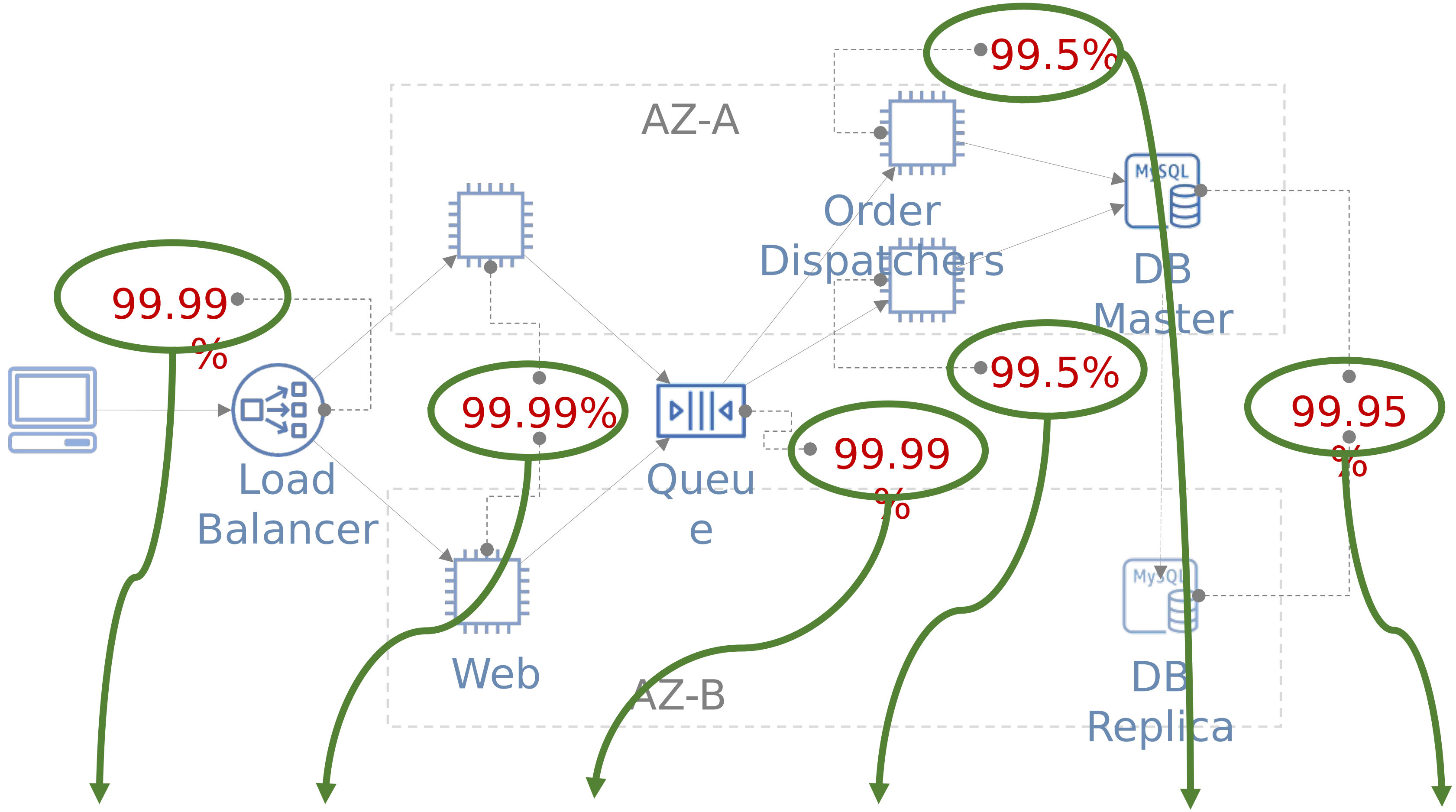
A=99% S=2

$$f = \frac{5!}{(2+1)! * (5-2-1)!} = 10$$

$$Availability_{system} \approx 1-10*(1-0.99)^{2+1} = 99.999\%$$

A=99%	Number of nodes the system tolerates to failure				
# of nodes	1	2	3	4	5
2	99,99%				
3	99,97%	99,9999%			
4	99,94%	99,9996%	99,999999%		
5	99,90%	99,9990%	99,999995%	99,99999999%	
6	99,85%	99,9980%	99,999985%	99,99999994%	99,9999999999%
7	99,79%	99,9965%	99,999965%	99,99999979%	99,9999999993%

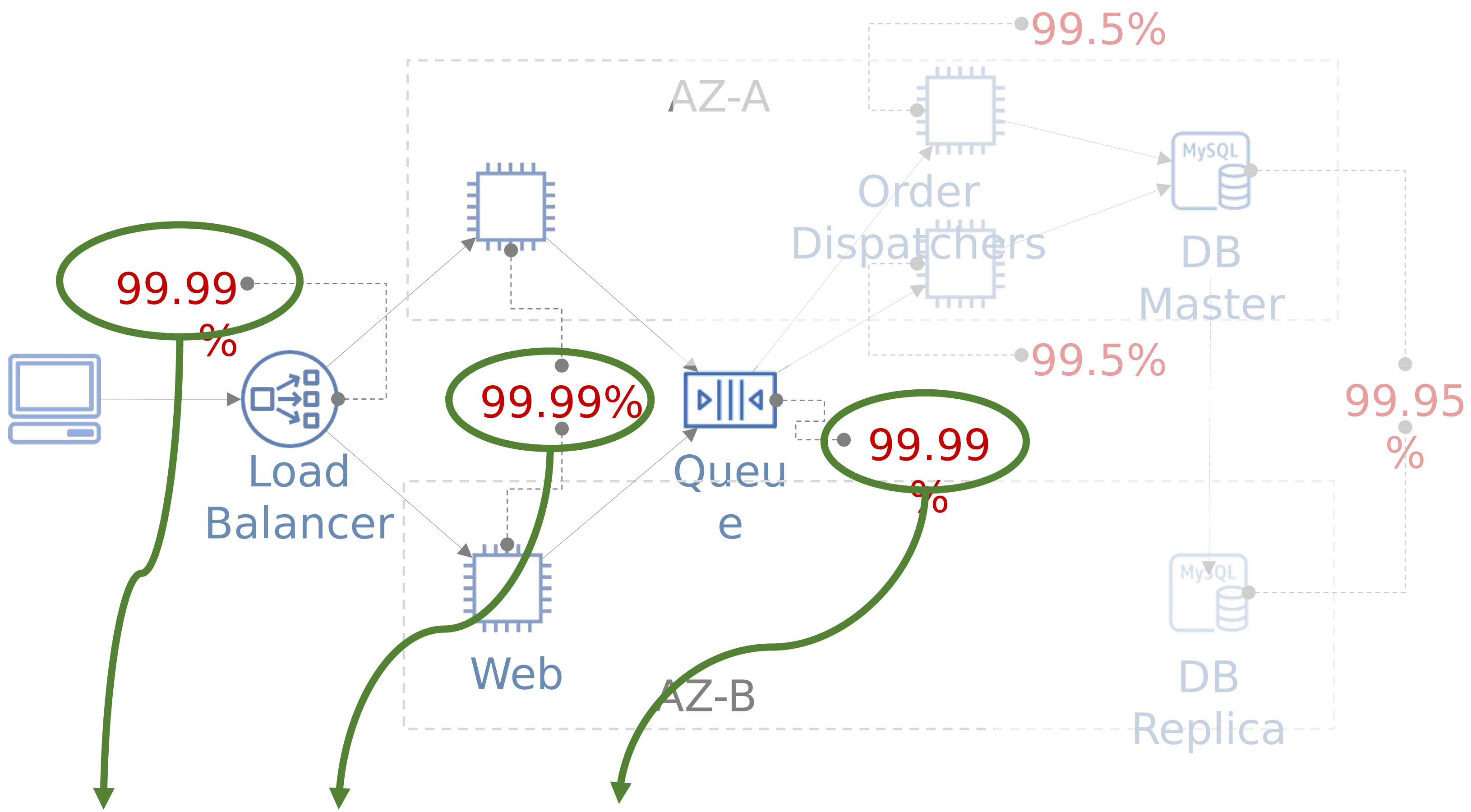
A1,2 – Availability o… …mber of spares, f – number of failure m…



bility = 0.9999*0.9999*0.9999*(1-(1-0.995)*(1-0.995))*0.9995 = 99.

?


Un-decouple



ailability = 0.9999*0.9999*0.9999 = 99.97%

Time definition

Amazon EC2 example

$$\text{Availability} = \frac{\text{Uptime}}{\text{Uptime}+\text{Downtime}}$$

- "Instance-Level Uptime Percentage" is calculated by subtracting from 100% the percentage of minutes during the month in which a Single EC2 Instance was in the state of Unavailability.
- "Unavailable" and "Unavailability" mean:
  - For the Instance-Level SLA, your Single EC2 Instance has no external connectivity.

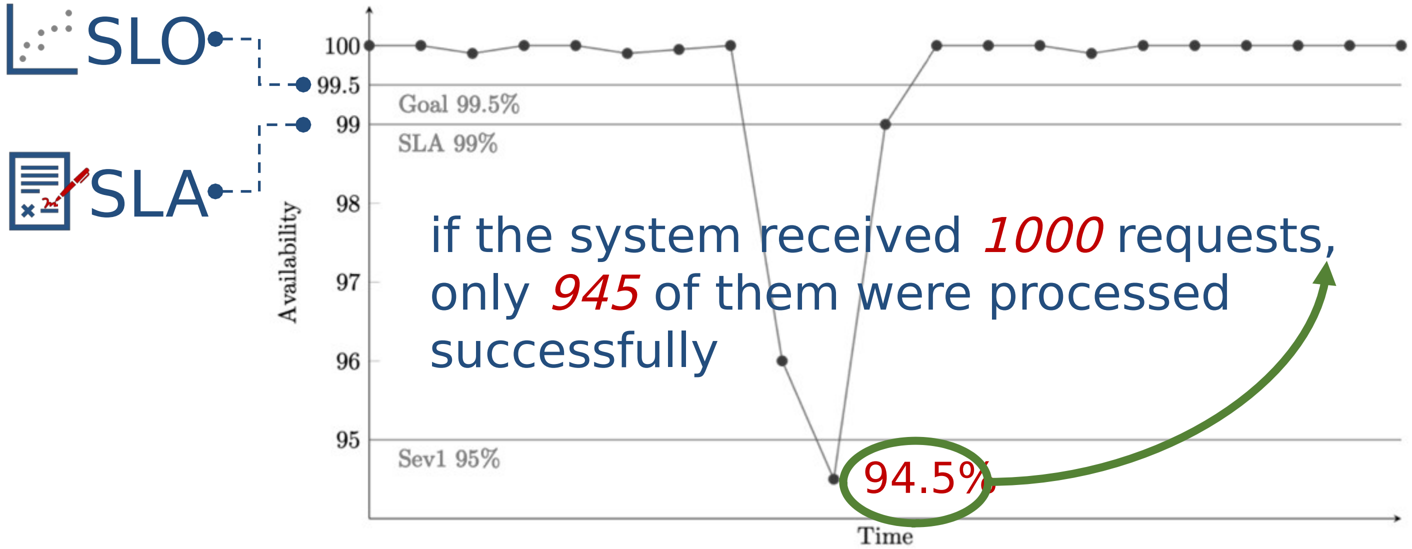
Availability	Monthly Unavailability	Application categories
99%	7.2 hrs.	Batch processing, data extraction, transfer and load jobs
99.9%	43 min.	Knowledge management, project tracking
99.95%	22 min.	Online commerce
99.99%	4 min.	Video delivery, broadcast streaming

Error Rate definition

Amazon SQS example

$$\text{Availability} = \frac{\text{Successfully Processed Units of Work}}{\text{Total Valid Units of Work Received}}$$

- "Availability" is calculated for each 5-minute interval as the percentage of Requests processed by the applicable Included Service that do not fail with Errors and relate solely to the provisioned Included Service. If you did not make any Requests in a given 5-minute interval, that interval is assumed to be 100% available.
- An "Error" is any Request that returns a 500 or 503 error code.



Which definition is better?

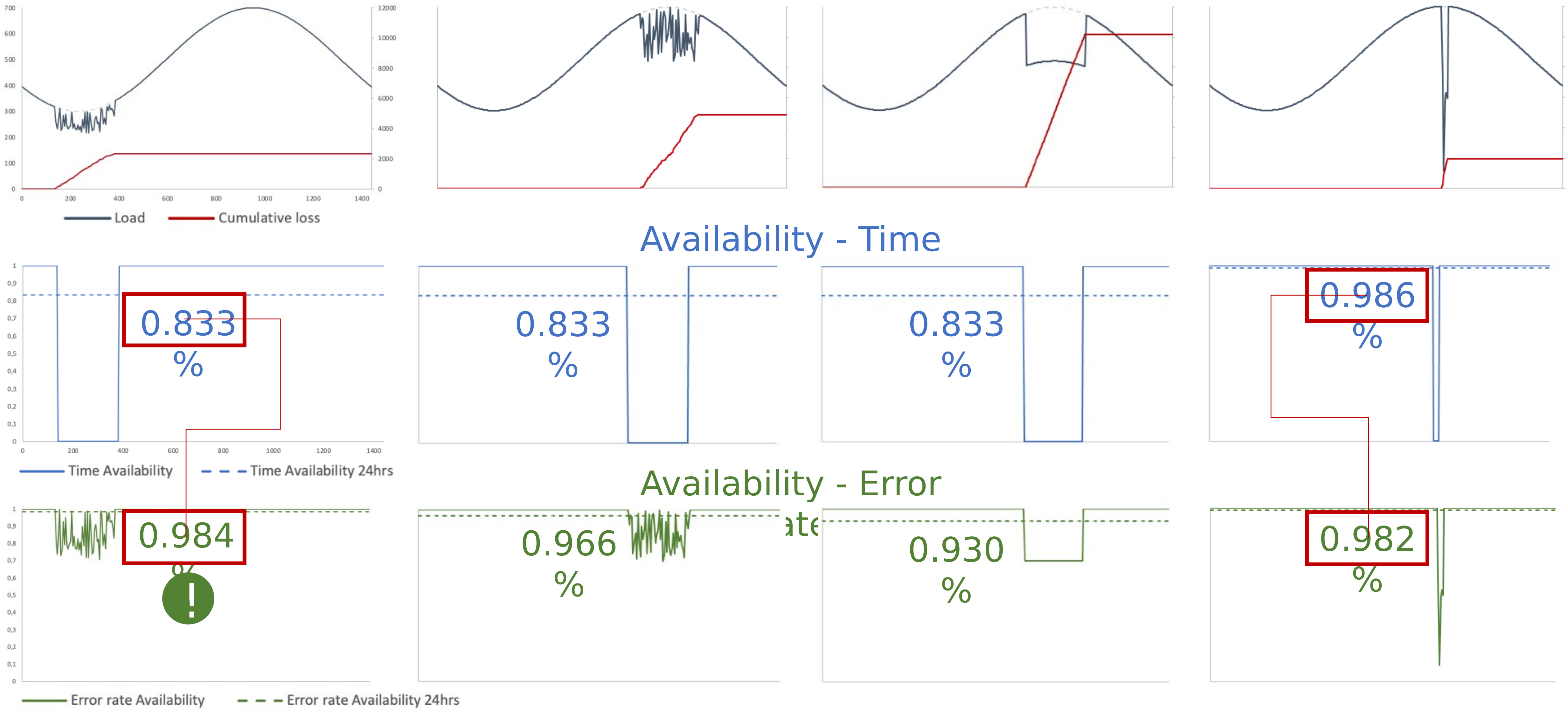
Amazon services example

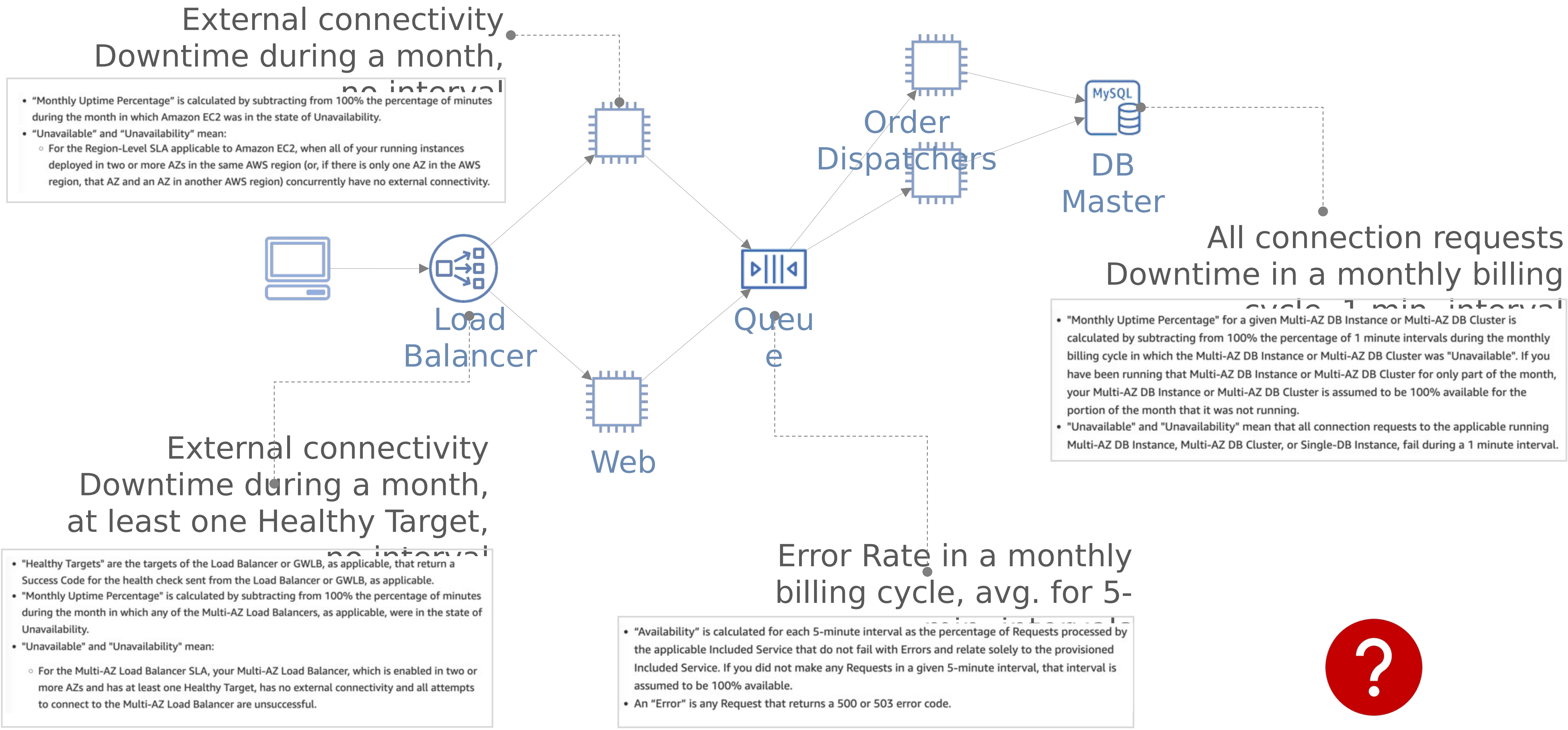
	Service	Platform
aws	Time Error Rate	Error Rate
amazon alexa	Time	
amazon.com	Error Rate	Time
amazon advertising	Time	
prime video	Time	Time


Enlarge

Error rate is generally higher

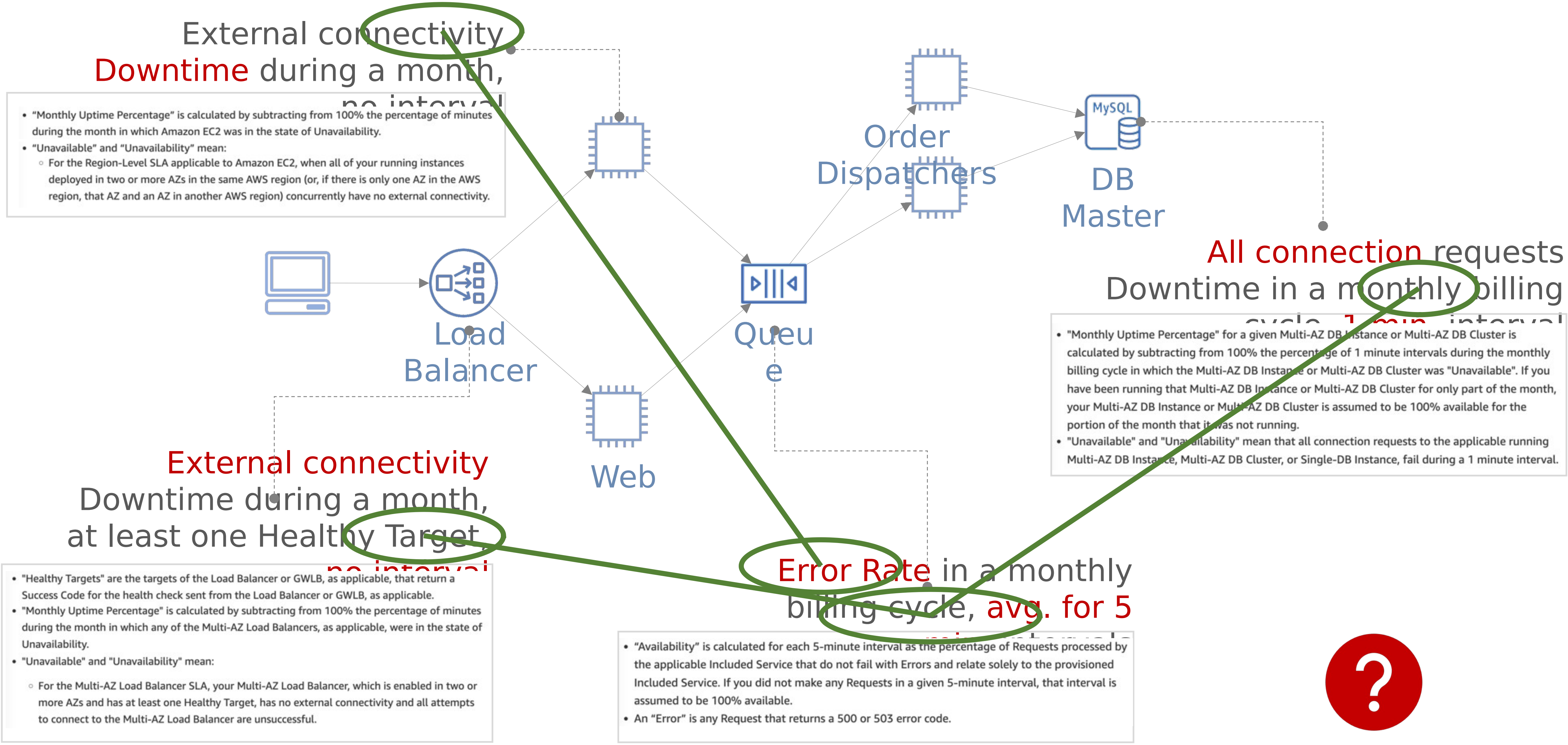
Low long | Low long peak | Constant long peak | High short





External connectivity
Downtime during a month,
• "Monthly Uptime Percentage" is calculated by subtracting from 100% the percentage of minutes during the month in which Amazon EC2 was in the state of Unavailability.
• "Unavailable" and "Unavailability" mean:
◦ For the Region-Level SLA applicable to Amazon EC2, when all of your running instances deployed in two or more AZs in the same AWS region (or, if there is only one AZ in the AWS region, that AZ and an AZ in another AWS region) concurrently have no external connectivity.
Order
Dispatchers
MySQL
DB
Master
All connection requests
Downtime in a monthly billing
• "Monthly Uptime Percentage" for a given Multi-AZ DB Instance or Multi-AZ DB Cluster is calculated by subtracting from 100% the percentage of 1 minute intervals during the monthly billing cycle in which the Multi-AZ DB Instance or Multi-AZ DB Cluster was "Unavailable". If you have been running that Multi-AZ DB Instance or Multi-AZ DB Cluster for only part of the month, your Multi-AZ DB Instance or Multi-AZ DB Cluster is assumed to be 100% available for the portion of the month that it was not running.
• "Unavailable" and "Unavailability" mean that all connection requests to the applicable running Multi-AZ DB Instance, Multi-AZ DB Cluster, or Single-DB Instance, fail during a 1 minute interval.
Load
Balancer
Queu
e
Web
External connectivity
Downtime during a month,
at least one Healthy Target,
• "Healthy Targets" are the targets of the Load Balancer or GWLB, as applicable, that return a Success Code for the health check sent from the Load Balancer or GWLB, as applicable.
• "Monthly Uptime Percentage" is calculated by subtracting from 100% the percentage of minutes during the month in which any of the Multi-AZ Load Balancers, as applicable, were in the state of Unavailability.
• "Unavailable" and "Unavailability" mean:
◦ For the Multi-AZ Load Balancer SLA, your Multi-AZ Load Balancer, which is enabled in two or more AZs and has at least one Healthy Target, has no external connectivity and all attempts to connect to the Multi-AZ Load Balancer are unsuccessful.
Error Rate in a monthly
billing cycle, avg. for 5-
• "Availability" is calculated for each 5-minute interval as the percentage of Requests processed by the applicable Included Service that do not fail with Errors and relate solely to the provisioned Included Service. If you did not make any Requests in a given 5-minute interval, that interval is assumed to be 100% available.
• An "Error" is any Request that returns a 500 or 503 error code.
?

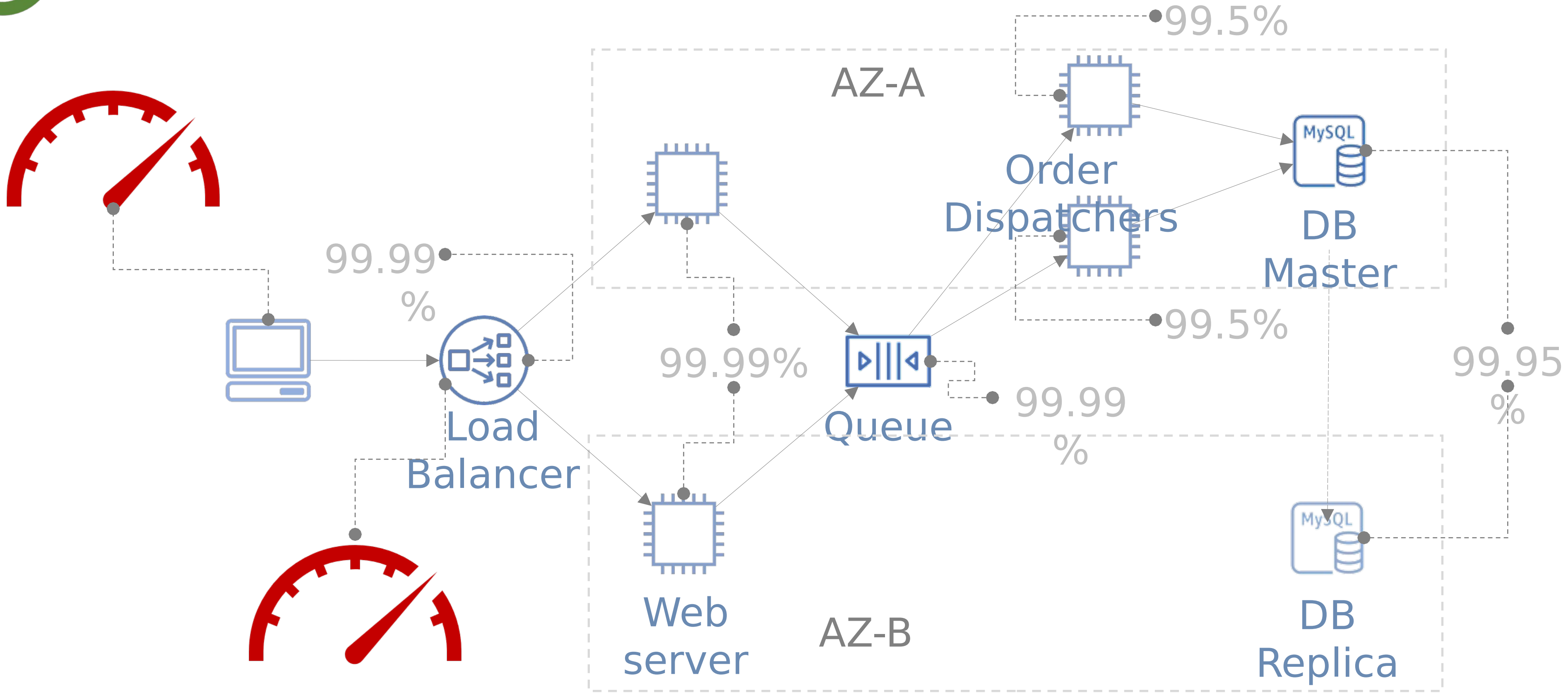

😈unt uncountable
EVIDENCE
LC



External connectivity
Downtime during a month,
no interval
• "Monthly Uptime Percentage" is calculated by subtracting from 100% the percentage of minutes during the month in which Amazon EC2 was in the state of Unavailability.
• "Unavailable" and "Unavailability" mean:
◦ For the Region-Level SLA applicable to Amazon EC2, when all of your running instances deployed in two or more AZs in the same AWS region (or, if there is only one AZ in the AWS region, that AZ and an AZ in another AWS region) concurrently have no external connectivity.
Order
Dispatchers
MySQL
DB
Master
All connection requests
Downtime in a monthly billing
• "Monthly Uptime Percentage" for a given Multi-AZ DB Instance or Multi-AZ DB Cluster is calculated by subtracting from 100% the percentage of 1 minute intervals during the monthly billing cycle in which the Multi-AZ DB Instance or Multi-AZ DB Cluster was "Unavailable". If you have been running that Multi-AZ DB Instance or Multi-AZ DB Cluster for only part of the month, your Multi-AZ DB Instance or Multi-AZ DB Cluster is assumed to be 100% available for the portion of the month that it was not running.
• "Unavailable" and "Unavailability" mean that all connection requests to the applicable running Multi-AZ DB Instance, Multi-AZ DB Cluster, or Single-DB Instance, fail during a 1 minute interval.
Load
Balancer
Queu
e
Web
External connectivity
Downtime during a month,
at least one Healthy Target
• "Healthy Targets" are the targets of the Load Balancer or GWLB, as applicable, that return a Success Code for the health check sent from the Load Balancer or GWLB, as applicable.
• "Monthly Uptime Percentage" is calculated by subtracting from 100% the percentage of minutes during the month in which any of the Multi-AZ Load Balancers, as applicable, were in the state of Unavailability.
• "Unavailable" and "Unavailability" mean:
◦ For the Multi-AZ Load Balancer SLA, your Multi-AZ Load Balancer, which is enabled in two or more AZs and has at least one Healthy Target, has no external connectivity and all attempts to connect to the Multi-AZ Load Balancer are unsuccessful.
Error Rate in a monthly
billing cycle, avg. for 5
• "Availability" is calculated for each 5-minute interval as the percentage of Requests processed by the applicable Included Service that do not fail with Errors and relate solely to the provisioned Included Service. If you did not make any Requests in a given 5-minute interval, that interval is assumed to be 100% available.
• An "Error" is any Request that returns a 500 or 503 error code.
?

Where to get the number

Measure



- Measure consumer's and provider’s SLI
- Define reasonable SLO


😈ompare incomparable

Mix up comparison

Vendor 1

Uptime Percentage = (Maximum Service Uptime - number of minutes of Unavailability) / Maximum Service Uptime
| Unavailability = 50x response codes for more than ***10%*** of queries during at least ***5min***

Vendor 2

Uptime Percentage = 100% - average(Error Rates measured over each ***5min*** period during the calendar month)
| Error Rate = number of Valid Requests with an HTTP Status in the 500-range / total number of Valid Requests.
Repeated identical requests do not count towards the Error Rate unless they conform to the Back-off Requirements
| Back-off Requirements = when an error occurs, an Application is responsible for waiting for a period of time
before issuing another request. The minimum back-off interval is ***1sec*** and for each consecutive error, the back-
off interval increases exponentially ***up to 32 seconds***

Vendor 3

Vendor 4

Uptime Percentage = 100% - Average Error Rate
| Average Error Rate = sum(Error Rates for each hour in the ***billing month*** / total number of hours
| Error Rate = total number of Failed Transactions / Total Transactions during ***one hour***
| Failed Transactions = transactions that are not completed within the Maximum

Take responsibility

Commit for penalties

Amazon SQS and SNS example

SLA

Service Commitment

AWS will use commercially reasonable efforts to make the Included Services each available with a Monthly Uptime Percentage for each AWS region, during any monthly billing cycle, of at least 99.9% (the "Service Commitment"). In the event any of the Included Services do not meet the Service Commitment, you will be eligible to receive a Service Credit as described below.

Service Credits

Service Credits are calculated as a percentage of the total charges paid by you for the applicable Included Service in the applicable AWS region for the monthly billing cycle in which the Monthly Uptime Percentage fell within the ranges set forth in the table below:

Monthly Uptime Percentage	Service Credit Percentage
Less than 99.9% but greater than or equal to 99.0%	10%
Less than 99.0% but greater than or equal to 95.0%	25%
Less than 95.0%	100%

Penalties

Do not pay

y just a fraction for failing your commitment

Credit amount is expressed as a *percentage of the service's monthly bill* that did not meet SLA. It will be credited to future monthly bills.

Vendor 1

Monthly uptime	Service Credits
99.5%-99.0%	10%
99.0%-95%	30%
< 95.0%	100%
> 6 minutes	100%

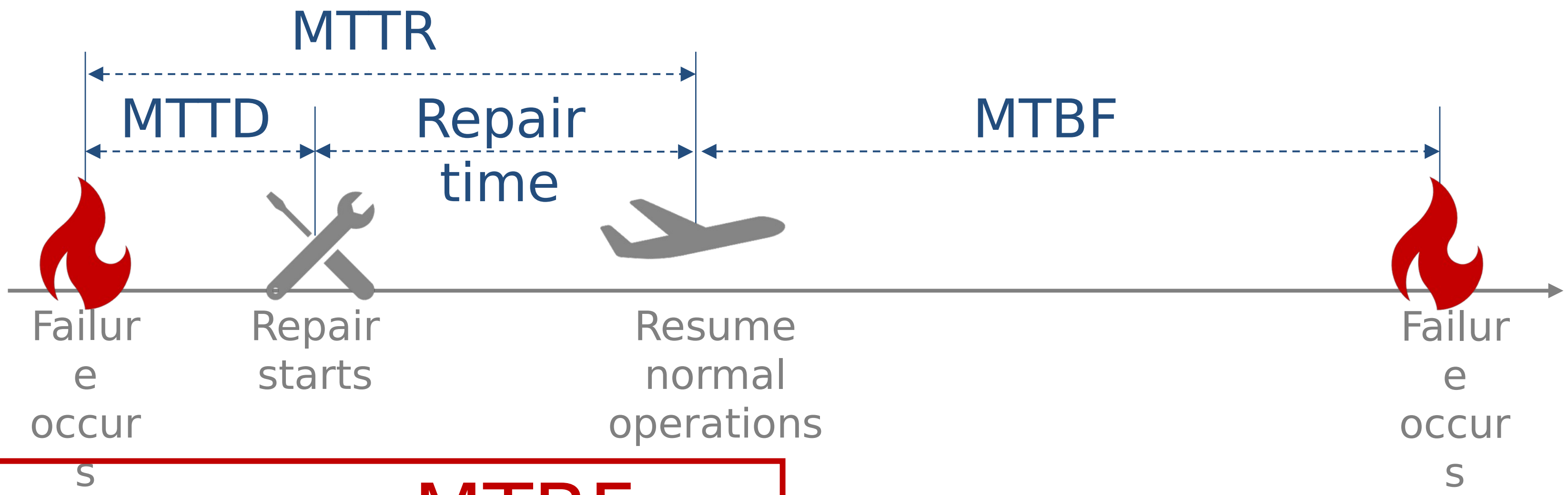
Vendor 2

Monthly uptime	Service Credits
99.5%-95.0%	10%
95.0%-90.0%	25%
< 90.0%	100%
< 1 minute	not counted

Vendor 3

Monthly uptime	Service Credits
99.9%-99.0%	5%
98.99%-90%	10%
< 90.0%	20%
< 3 minutes	not counted

Mean Time definition

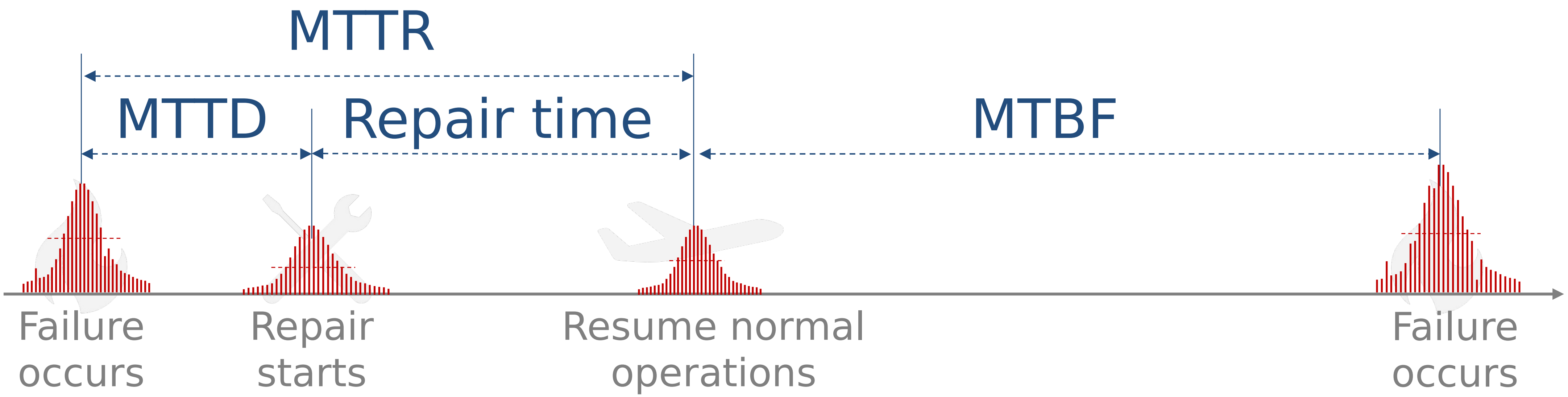


$$\boxed{\text{Availability} = \frac{\text{MTBF}}{\text{MTBF}+\text{MTTR}}}$$

- Mean time between failure (MTBF) - an average time between when a system begins normal operation and its next failure
- Mean time to repair (MTTR) - a period of time when the system is unavailable while the failed component is returned to service
- Mean time to detection (MTTD) - an amount of time between a failure occurring and when repair operations begin


😈redict and average

Quantitative instead of qualitative



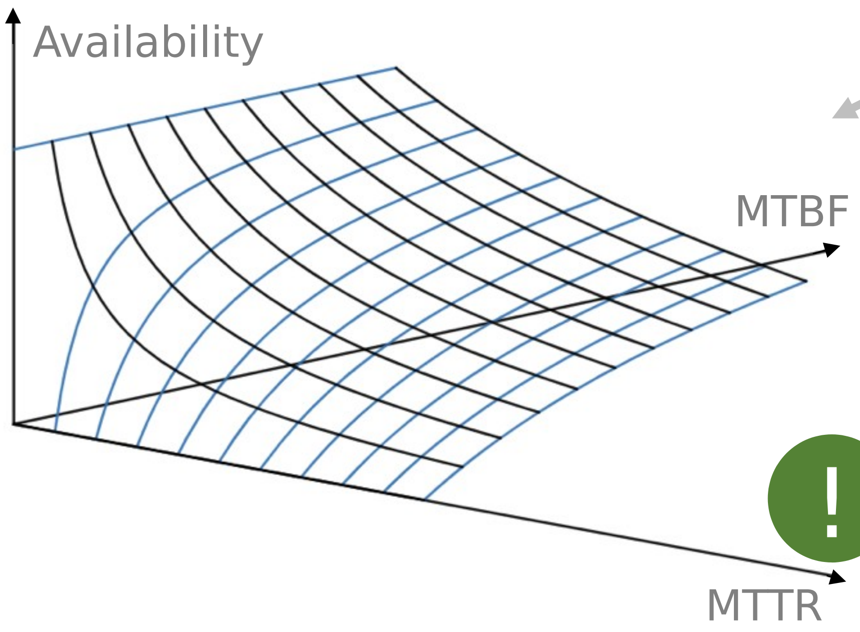
MTxx

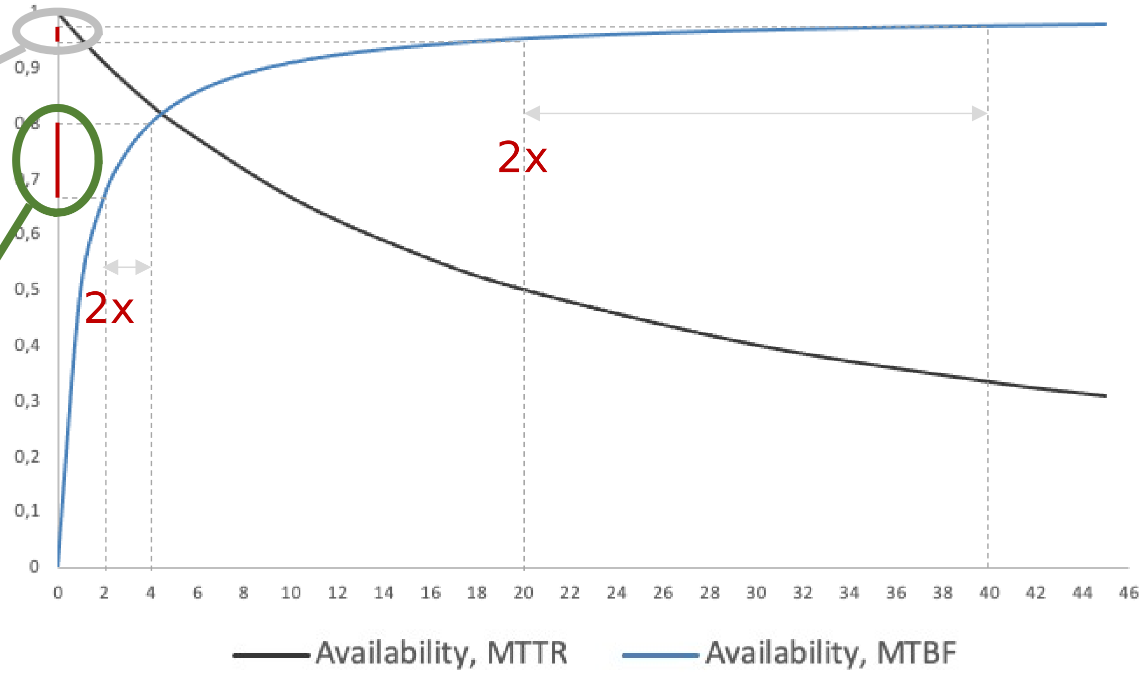
- are always derived from prediction or forecast
- are averages

Focus on MTTR

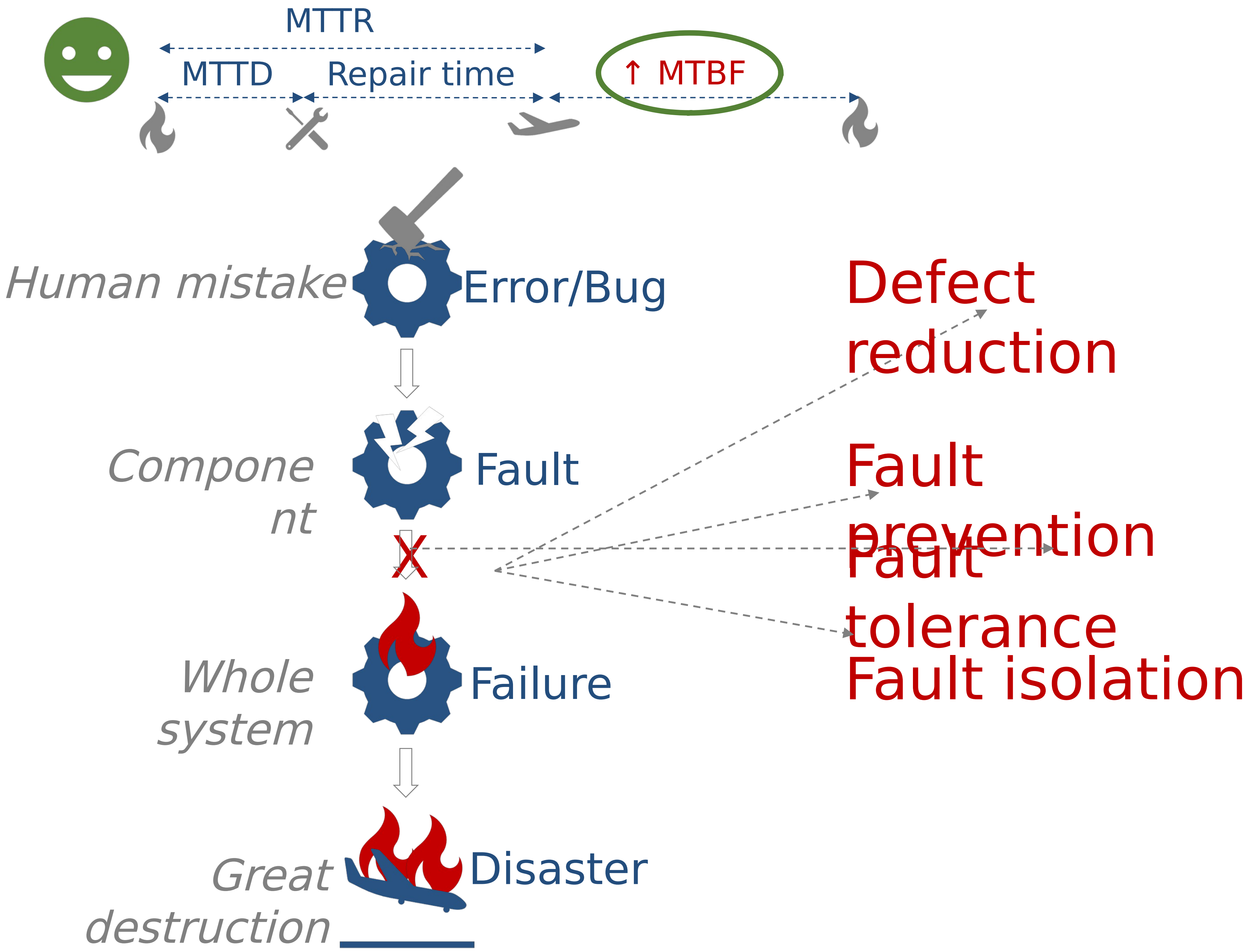
5-second rule :)

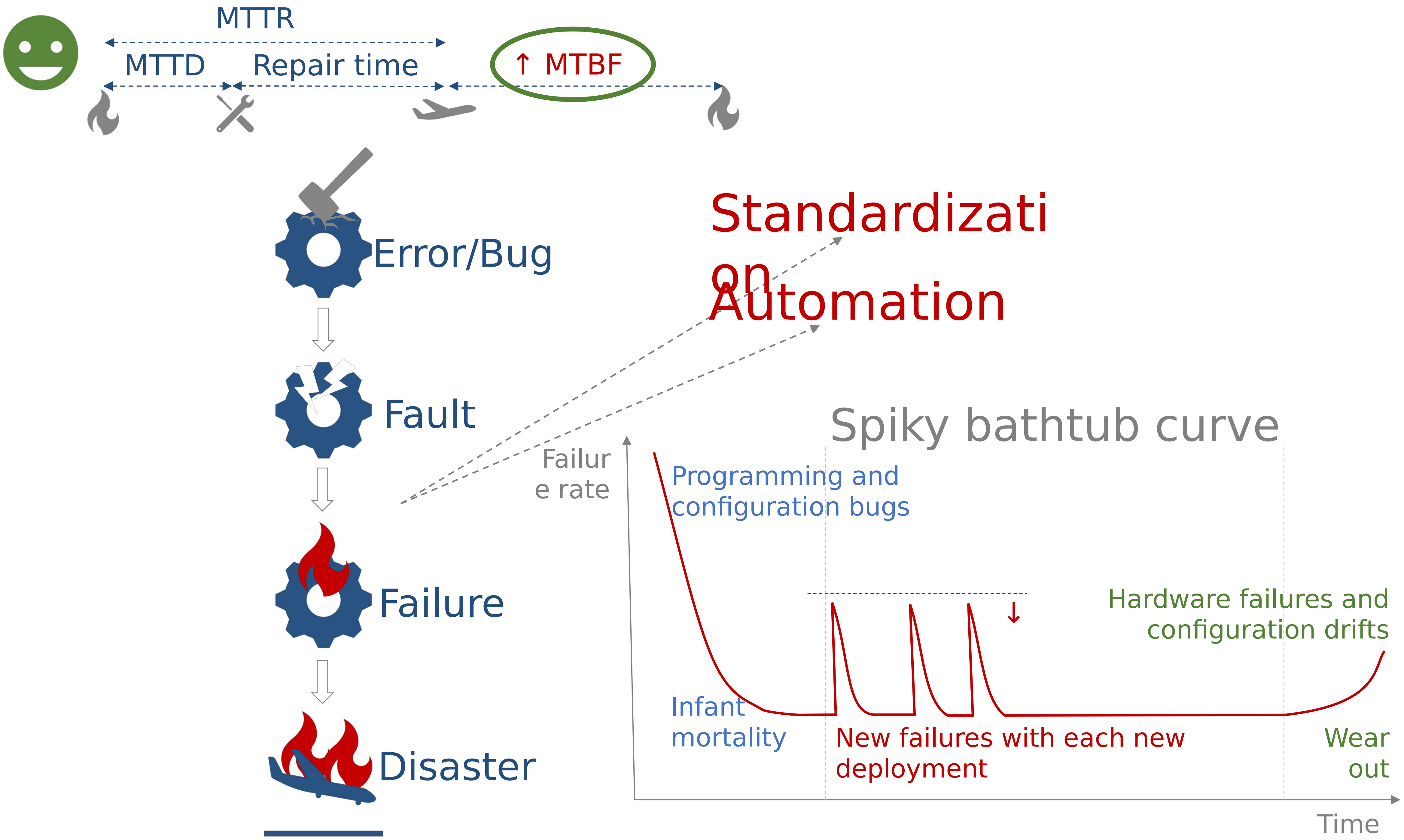
$$\text{Availability} = \frac{\text{MTBF}}{\text{MTBF}+\text{MTTR}}$$

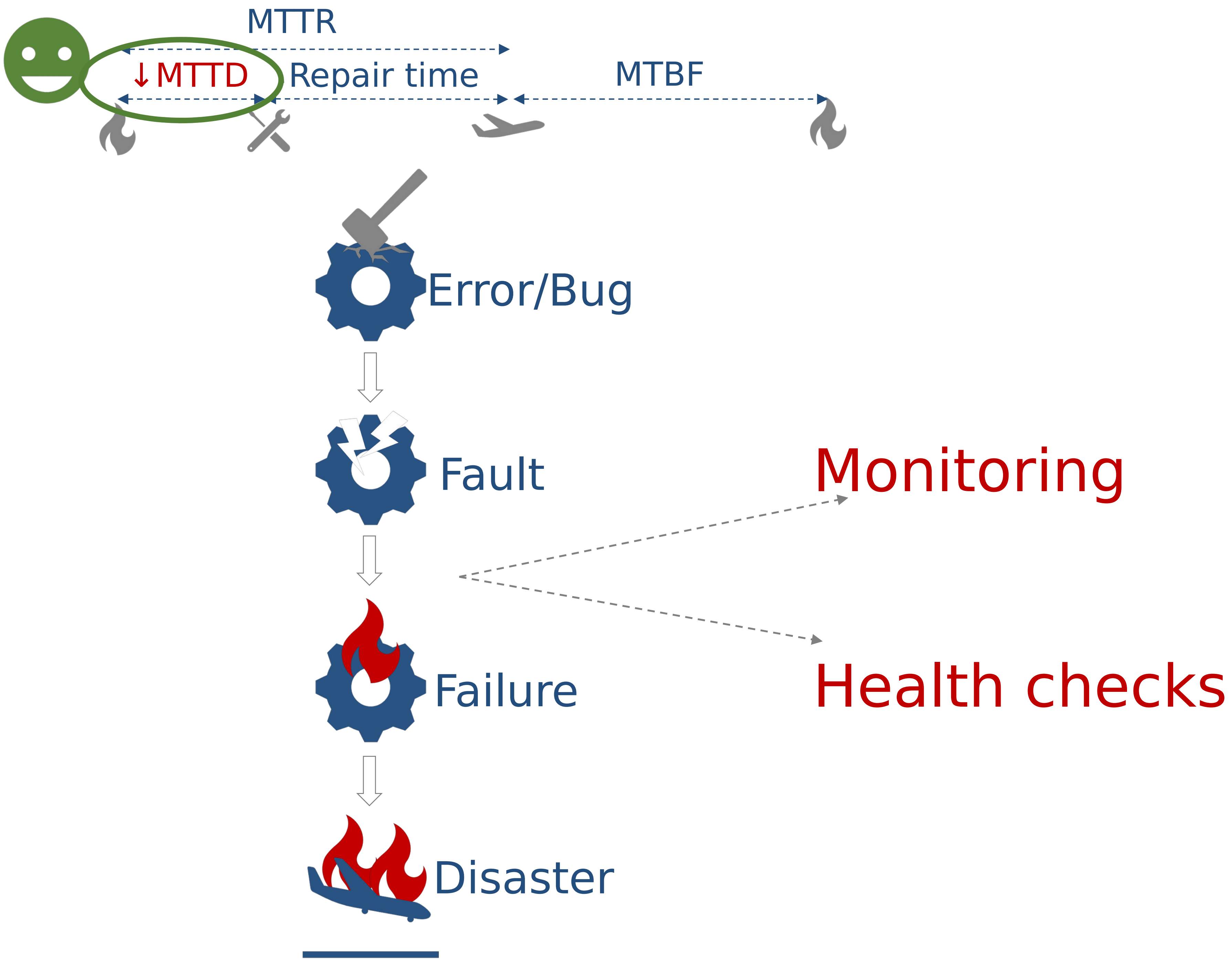


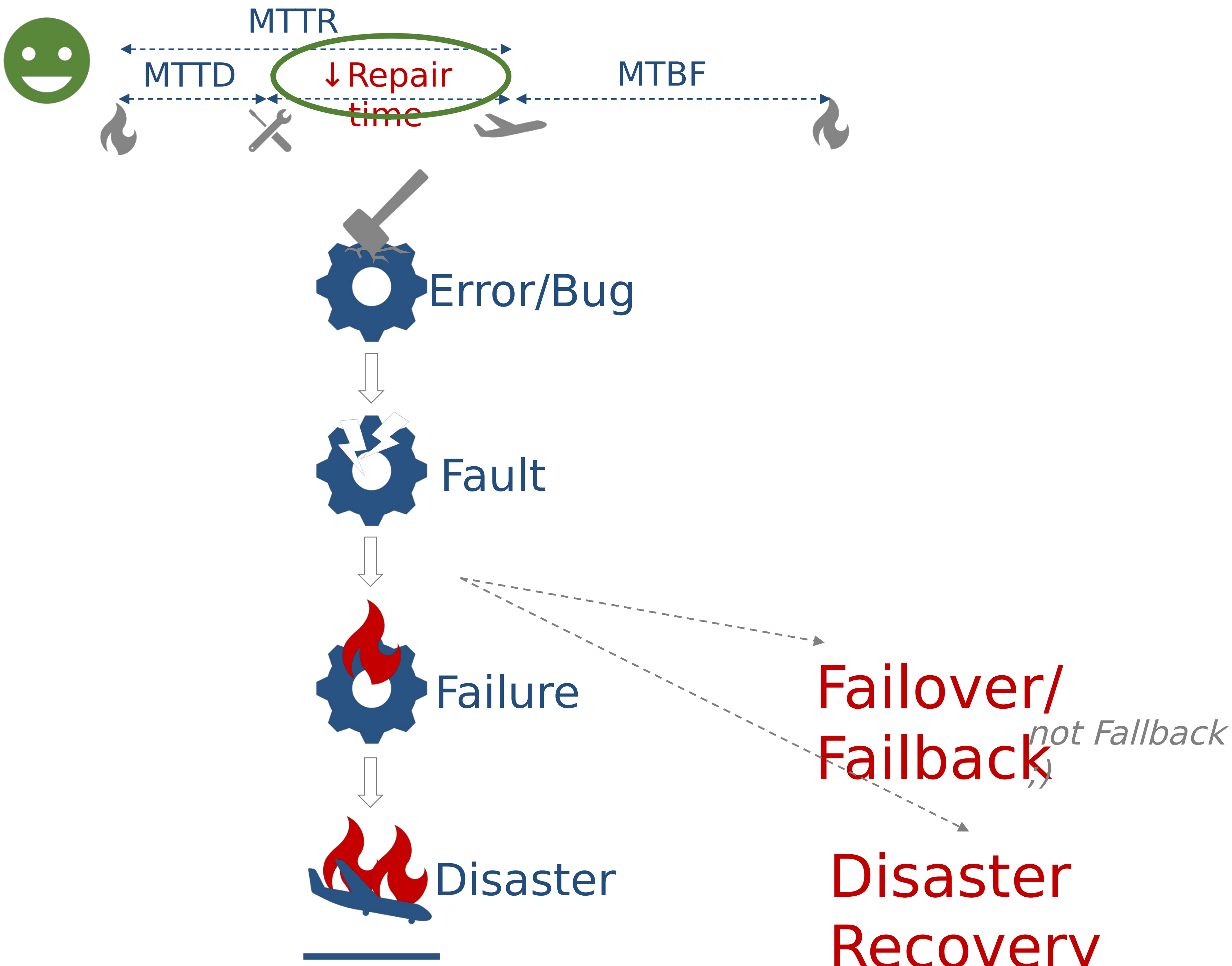


But… resource that often goes down (MTBF) and recovers quickly (MTTR) might trigger expensive failure handling


MTTR
MTTD
Repair time
↑ MTBF
Human mistake
Error/Bug
Defect reduction
Compone
nt
Fault
Fault prevention
Fault tolerance
X
Whole system
Failure
Fault isolation
Great destruction
Disaster


MTTR
MTTD
Repair time
↑ MTBF
Error/Bug
Fault
Failure
Disaster
Standardization
Automation
Spiky bathtub curve
Failure rate
Programming and configuration bugs
Hardware failures and configuration drifts
Infant mortality
New failures with each new deployment
Wear out
Time


MTTR
↓MTTD
Repair time
MTBF
Error/Bug
Fault
Failure
Disaster
Monitoring
Health checks


MTTR
MTTD
↓Repair time
MTBF
Error/Bug
Fault
Failure
Disaster
Failover/
Failback
not Fallback ;)
Disaster Recovery

Check other ways

Must read

Neither of these definitions are adequate. The first is unsatisfactory because the design and deployment of cloud systems, such as Azure [14], Dynamo [16], or Gmail, actively avoid single points of failure by sharding data across many machines and using replication [23] with failover when problems occur. Consequently, these systems rarely have an out-

metrics to quantify service reliability [10, 11]. A good availability metric should be *meaningful*, *proportional*, and *actionable*. By "meaningful" we mean that it should capture what users experience. By "proportional" we mean that a change in the metric should be proportional to the change in user-perceived availability. By "actionable" we mean that the metric should give system owners insight into why availability for a period was low. This paper shows that none

Meaningful Availability

Tamás Hauer, *Google* — Philipp Hoffmann, *Google* — John Lunney, *Google* — Dan Ardelean, *Google* — Amer Diwan, *Google*

Abstract

High availability is a critical requirement for cloud applications: if a sytem does not have high availability, users cannot count on it for their critical work. Having a metric that meaningfully captures availability is useful for both users and system developers. It informs users what they should expect of the availability of the application. It informs developers what they should focus on to improve user-experienced availability. This paper presents and evaluates, in the context of Google's G Suite, a novel availability metric: windowed user-uptime. This metric has two main components. First, it directly models user-perceived availability and avoids the bias in commonly used availability metrics. Second, by simultaneously calculating the availability metric over many windows it can readily distinguish between many short periods of unavailability and fewer but longer periods of unavailability.

1 Introduction

Users rely on productivity suites and tools, such as G Suite, Office 365, or Slack, to get their work done. Lack of *availability* in these suites comes at a cost: lost productivity, lost revenue and negative press for both the service provider and the user [1, 3, 6]. System developers and maintainers use metrics to quantify service reliability [10, 11]. A good availability metric should be *meaningful*, *proportional*, and *actionable*. By "meaningful" we mean that it should capture what users experience. By "proportional" we mean that a change in the metric should be proportional to the change in user-perceived availability. By "actionable" we mean that the metric should give system owners insight into why availability for a period was low. This paper shows that none of the commonly used metrics satisfy these requirements and presents a new metric, *windowed user-uptime* that meets these requirements. We evaluate the metric in the context of Google's G Suite products, such as Google Drive and Gmail.

The two most commonly used approaches for quantifying availability are *success-ratio* and *incident-ratio*. Success-ratio is the fraction of the number of successful requests to total requests over a period of time (usually a month or a quarter) [5, 2, 9]. This metric has some important shortcomings. First, it is biased towards the most active users; G Suite's most active users are 1000x more active than its least active (yet still active) users. Second, it assumes that user behavior does not change during an outage, although it often does: e.g., a user may give up and stop submitting requests during an outage which can make the measured impact appear smaller than it actually is. Incident-ratio is the ratio of "up minutes" to "total minutes", and it determines "up minutes" based on the duration of known incidents. This metric is inappropriate for large-scale distributed systems since they are almost never completely down or up.

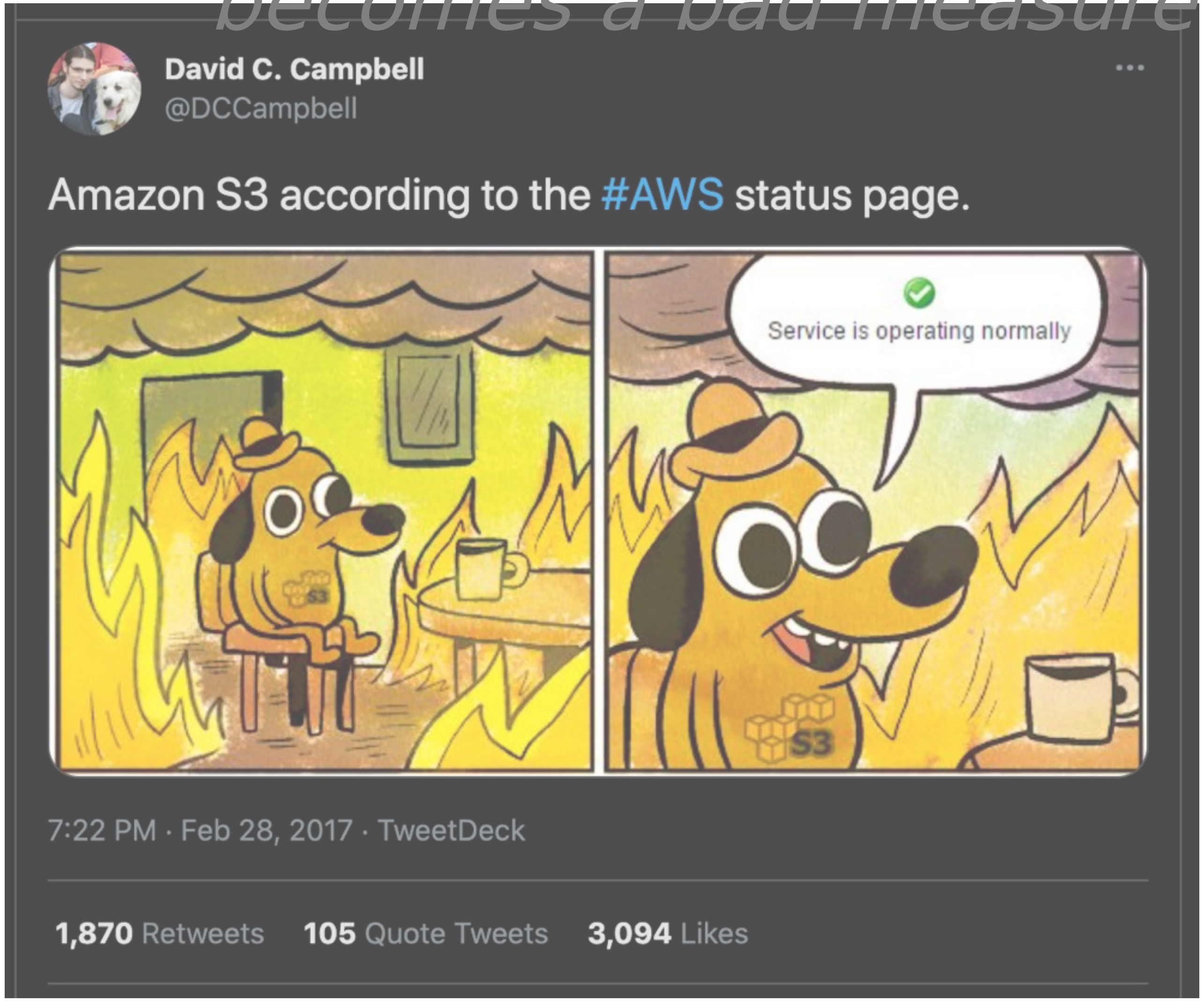
Our approach, *windowed user-uptime* has two components. First, user-uptime analyzes fine-grained user request logs to determine the up and down minutes for each user and aggregates these into an overall metric. By considering the failures that our users experience and weighing each user equally, this metric is meaningful and proportional. Second, windowed user-uptime simultaneously quantifies availability at all time windows, which enables us to distinguish many short outages from fewer longer ones; thus it enables our metric to be actionable.

We evaluate windowed user-uptime in the context of Google's G Suite applications and compare it to success-ratio. We show, using data from a production deployment of G Suite, that the above-mentioned bias is a real shortcoming of success-ratio and that windowing is an invaluable tool for identifying brief, but significant outages. Our teams systematically track down the root cause of these brief outages and address them before they trigger larger incidents.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 motivates the need for *windowed user-uptime* Section 4 describes user-uptime. Section 5 extends user-uptime with windowed user-uptime. Section 6 evaluates our approach and Section 8 concludes the paper.

Target

Goodhart's law: *every measure that becomes a target, becomes a bad measure*



Accept failure with post-mortem

aws.amazon.com/message/41926/

Summary of the Amazon S3 Service Disruption in the Northern Virginia (US-EAST-1) Region

We'd like to give you some additional information about the service disruption that occurred in the Northern Virginia (US-EAST-1) Region on the morning of February 28th, 2017. The Amazon Simple Storage Service (S3) team was debugging an issue causing the S3 billing system to progress more slowly than expected. At 9:37AM PST, an authorized S3 team member using an established playbook executed a command which was intended to remove a small number of servers for one of the S3 subsystems that is used by the S3 billing process. Unfortunately, one of the inputs to the command was entered incorrectly and a larger set of servers was removed than intended. The servers that were inadvertently removed supported two other S3 subsystems.

...

Finally, we want to apologize for the impact this event caused for our customers. While we are proud of our long track record of availability with Amazon S3, we know how critical this service is to our customers, their applications and end users, and their businesses. We will do everything we can to learn from this event and use it to improve our availability even further.

Reliability

	Availability	Durability
Objectives	Maximizing uptime	Minimizing suffer from data loss and corruption
Focus on	Components of the system	Data the system works with
Time	Mean values over a mid-term period	Long-term likelihood of data loss or corruption
Scale	Small-scale common disruptions	Small-scale corruption and large-scale loss problems


Hide

Durability and AFR

Durability - likelihood of avoiding data loss or corruption

Data health = the data you retrieved it is the same as you stored

$$\boxed{Durability_{system} = 1 - AFR_{system}}$$

Annualized Failure Rate (AFR) - probability of system failure during a year

Durability is the inverse of AFR, but in many cases, Durability is detached from AFR and calculated by itself. That is why it's defined in a broader scope than the annual rate.

Durability and AFR

AWS EBS and S3 example

Durability	Explanation
EBS gp3 volume AFR=0.2% Durability=99.8%	avg. loss of 0.2% of volumes over a given year or if you have 1,000 volumes running for one year, you should expect about 2 volumes to fail
S3 bucket AFR=0.000000001% Durability=99.999999999% (11x)	if you store 10,000 objects, on avg. you may lose one of them every 10,000,000 years or if you store 1,000,000,000 objects, on avg. you may lose one of them every 100 years


😈pply unapplicable

Refer to industry-accepted guidelines

For AFR, [illegible] uses industry-accepted guidelines and assumes a conservative drive AFR of 5%. In practice, our observed AFRs are much lower. For MTTR, [illegible] estimates 3.4 days per drive, based on the calculation below:

*MTTR = 14 TB drive capacity * 50 MB/s drive write speed = 3.4 days to fully write a replaced 14 TB drive*

As stated earlier in the erasure coding discussion, in order for an object to be lost, more than 4 drives in a [illegible] storage slice would have to fail. To understand the probability of this, the first step is to understanding the probability a single drive failing using the calculation below:

*Probability of a 1 drive failing = AFR (5% year) * MTTR (3.4 days) * (1/365 year/days) = 4.66 x 10^{-4}*

The next step would be to understand the probability of four drives failing while another drive in the storage slice was rebuilding. This would be a potential data loss scenario because five drives in a storage slice would not be available (one in rebuild mode, plus four new failures). To calculate this probability, this formula applies:

Probability of 4 drives failing = Probability of 1 drive failing (4.66 x 10^{-4}) to the 4th power = 4.7 x 10^{-14}

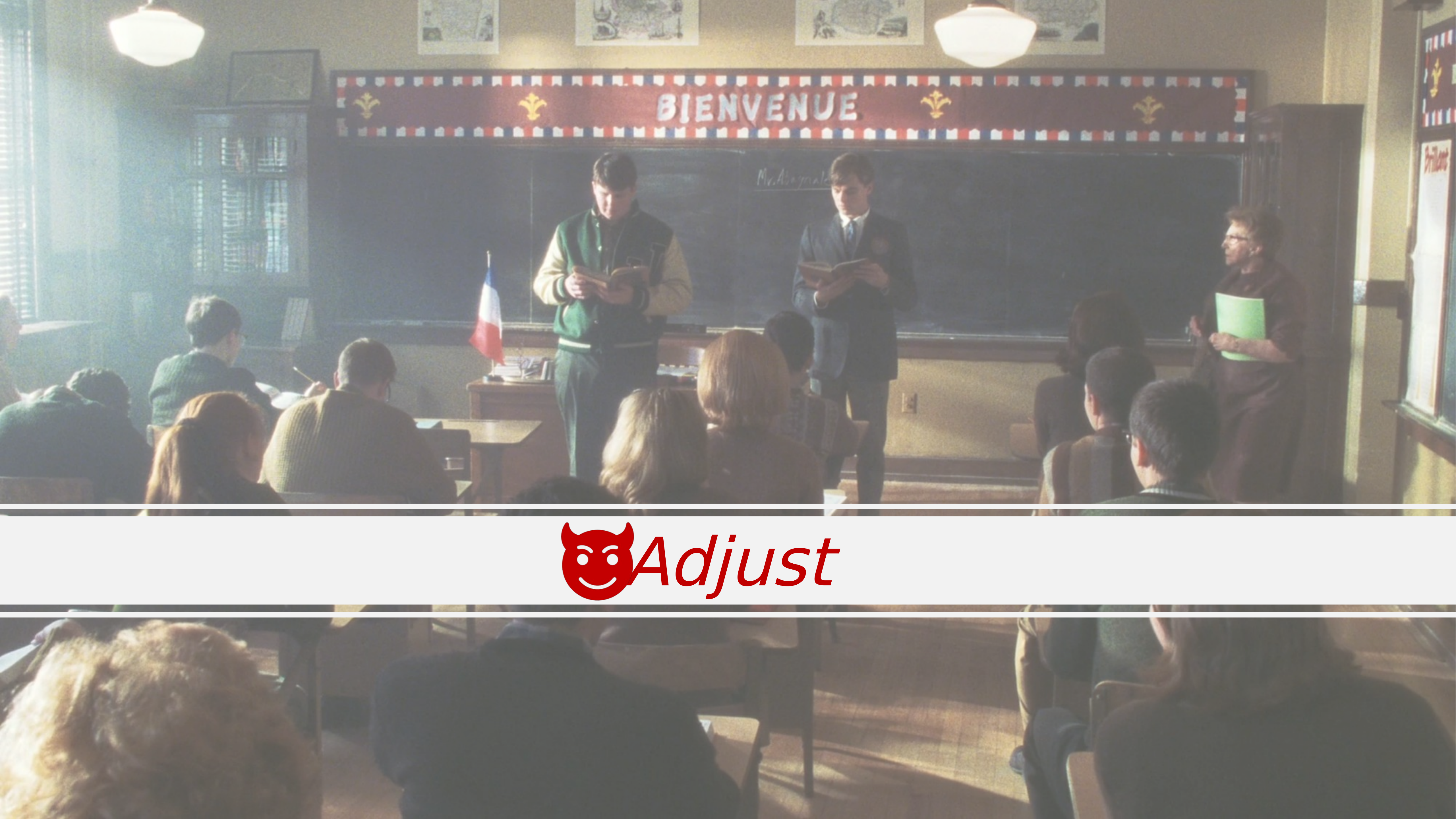
The final step in calculating data durability is to factor in the probability of 4 drives failing using the following formula:

Data Durability = 1 - (probability of 4 drives failing) = 1 - 4.7 x 10^{-14} = .99999999999995

As seen in the above calculations, [illegible] storage architecture provides greater than 11 x 9s of durability. The calculated number is actually 13 x 9s but for the sake of taking a conservative approach to the calculations and to align with how most of the hyperscalers position their data durability, [illegible] uses 11 x 9s as the published data durability metric.

hm… I calculated with different methods and got 7-10 x 9s

"Conservative" approach


BIENVENUE
Adjust

Undercovered calculation is adjusted to a common value

ate Lyve cloud S3 example

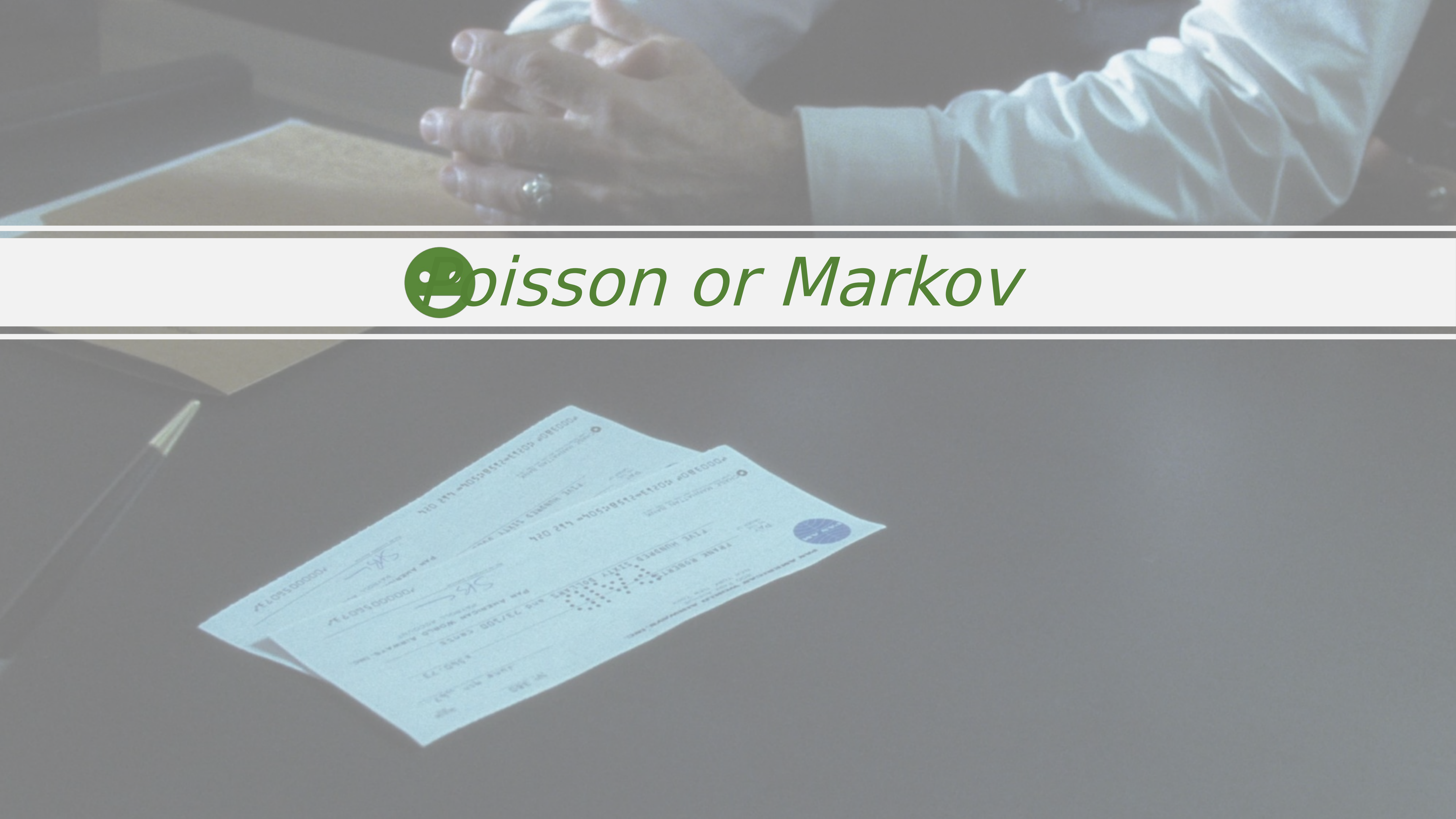
The implementation of **predictive failure algorithms** in Seagate HDDs and advanced SDC correction elevate Lyve Cloud's data durability. When further combined with advanced erasure code, where storage nodes are distributed across multiple racks and systems, Lyve Cloud takes data durability of public cloud S3-compatible storage to a new level, nearly eliminating the probability of data loss.

By building on the probability model used by others and applying to that our data protection algorithms, Lyve Cloud's theoretical data durability is **29 nines** (0.99999999999999999999999999999). We state a **conservative 11 nines** (0.99999999999) durability to match the industry standard offered by other providers.

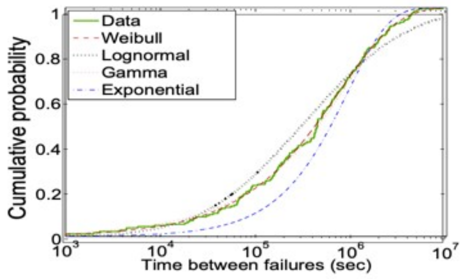
No way to check that theoretical calculation

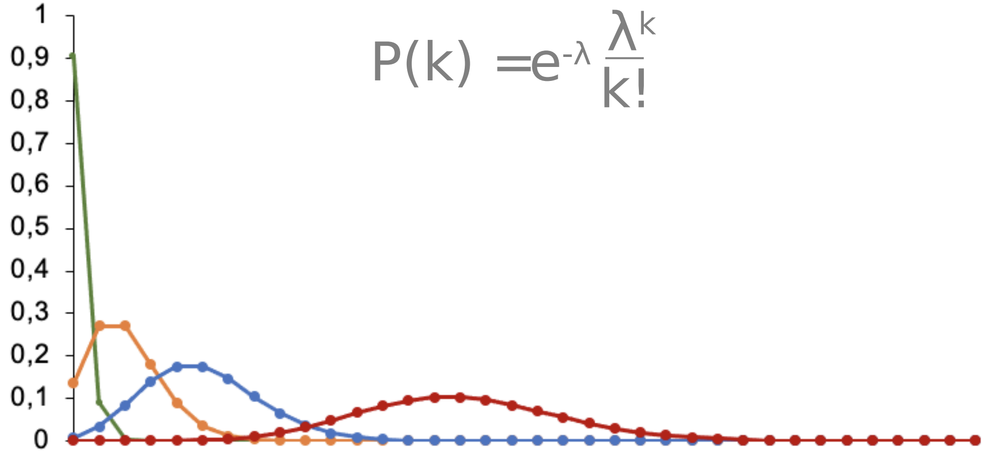
Just adjusted to the expected number

Poisson or Markov



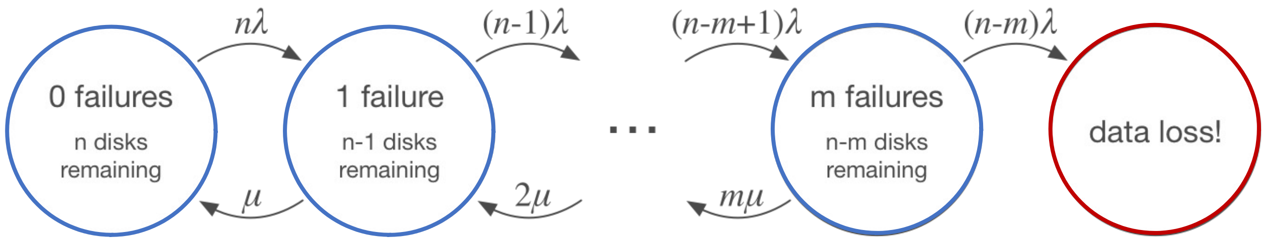
Start with Poisson distribution or Markov chain





$$P(k) = e^{-\lambda}\frac{\lambda^k}{k!}$$

$$R(n,m) = \frac{n!}{m!(n-m-1)!}\frac{\lambda^{m+1}}{\mu^m}$$

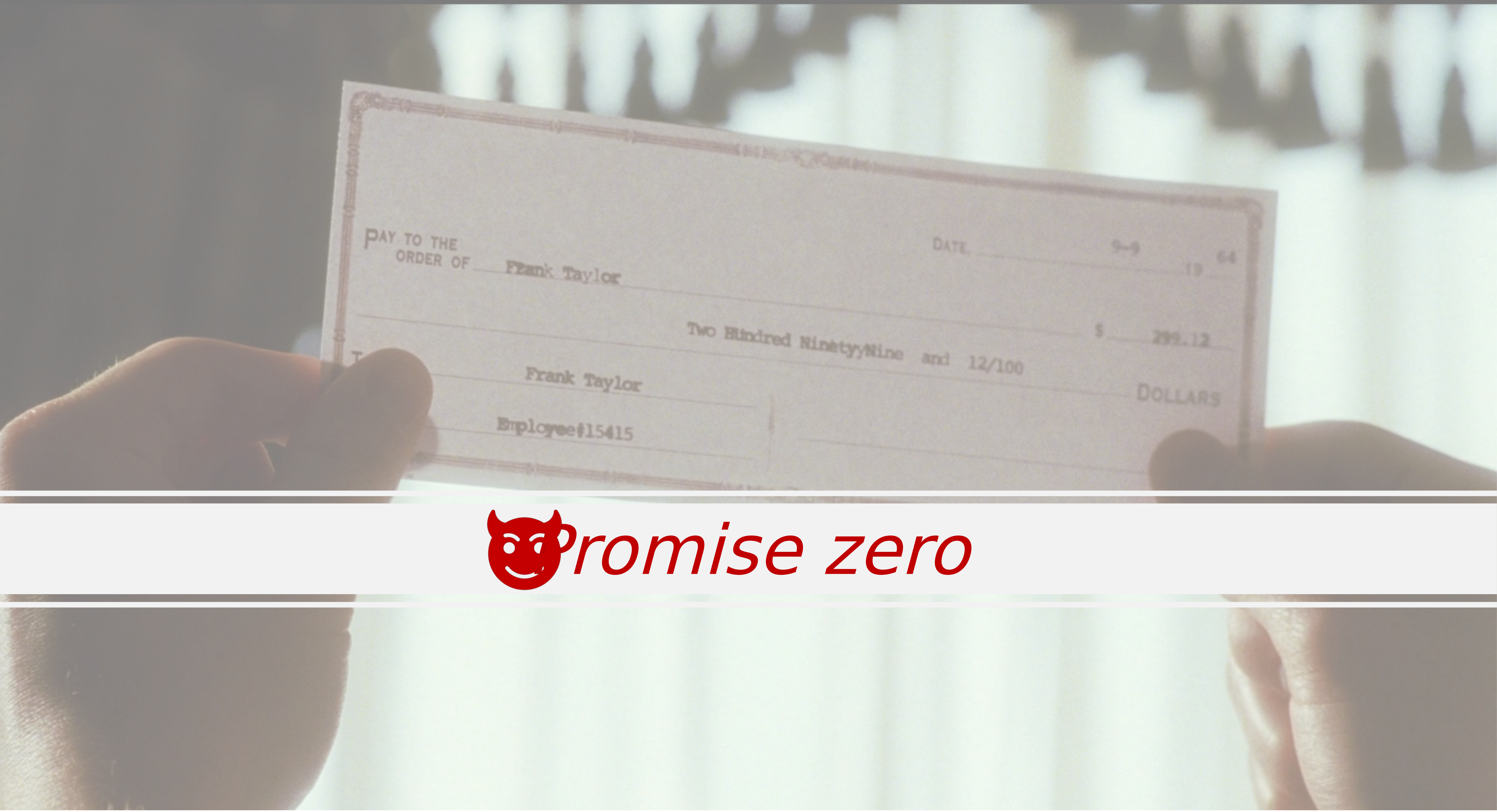


AFR	0,8%
MTTR, hrs	48
# of disks, n	20
# of failures to lose, k	4
Failure rate, λ	0,000876712
Interval durability	9,9999999999998E-01
Annual durability	0,999999999996

11x 9s 6

AFR	0,8%
MTTR, hrs	48
# of disks, n	20
# of failures before loss, m	3
MTBF, hrs	1091361
Failure rate, λ	9,16287E-07
Recovery rate, □	0,020833333
Data loss rate per hr	1,51E-15
Annual durability	0,99999999999

11x 9s



Promise zero

Zero failure rate

Azure managed disks example

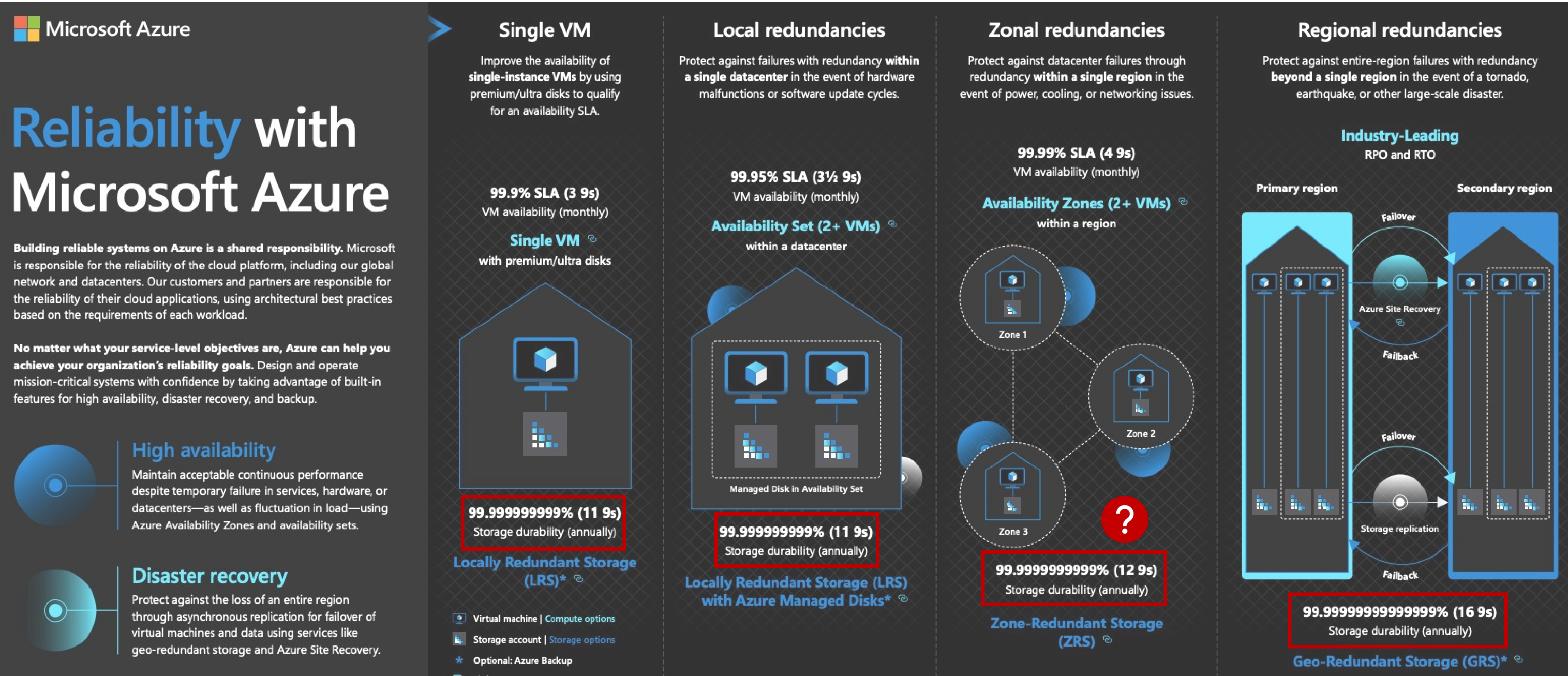
Managed disks are designed for 99.999% availability. Managed disks achieve this by providing you with three replicas of your data, allowing for high durability. If one or even two replicas experience issues, the remaining replicas help ensure persistence of your data and high tolerance against failures. This architecture has helped Azure consistently deliver enterprise-grade durability for infrastructure as a service (IaaS) disks, with an industry-leading ZERO% annualized failure rate.

I do not trust ZERO% failures


😈harge for nothing

A k to pay for 11+ nines *(12 nines in exceptional cases)*

e storage example



How to make a choice

Apples to apples -> one-to-one comparison

Amazon S3 and EBS example

	S3 standard	S3 Intelligent Tier	S3 Glacier Deep Archive
Designed for Durability	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)
Designed for Availability	99.99%	99.9%	99.99%
Availability SLA	99.9%	99%	99.9%

Same Durability but different Availability

	gp3	io2	io2 BE
Designed for Durability	99.8% - 99.9%	99.999%	99.999%
Designed for data plane Availability	99.999%	99.999%	99.999%
Designed for control plane Availability	99.95%	99.95%	99.95%
Instance level Availability SLA	99.5%	99.5%	99.5%

Same Availability but different Durability

If not sure...

Compare with

SCOT

SCOT

Subjective Competition Of Trusts

- Clear and simple definitions
- Fair penalties
- Shared SLOs & design principles
- Accepted failures & published post-mortems


Promise
&
Trust

ease leave your feedback

’s so important for us !